Escola Politècnica Superior d'Enginyeria de Manresa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE GRADO

Diseño y construcción de un puente grúa automatizado de precisión

Nombre: Daniel Andrade García
Especialidad: Ingeniería Electrónica Industrial y Automática
Tutor: Inmaculada Martínez Teixidor
Cotutor: Grau Baquero Armans

Convocatoria: Mayo 2015

Agradecimientos

Primeramente me gustaría agradecer muy sinceramente a la escuela de ingeniería de Igualada, por dejarme usar sus instalaciones y darme su ayuda en la medida de lo posible, aun no estando matriculado en ella.

En especial agradecer al profesor Grau Baquero Armans su ayuda respecto a la búsqueda de materiales, sistemas de mecanización y sobre todo a la creación de piezas con impresora 3D, sin él este proyecto no se podría haber llevado a cabo.

También dar las gracias a la ferretería Jorba Sola en general, por la ayuda prestada en la búsqueda de materiales y contactos de suministros.

Para finalizar, e indispensable el apoyo incondicional de la familia y los amigos, que con su apoyo e interés han hecho posible que no abandonase el proyecto y siempre mirase para adelante, con el fin de superar los obstáculos y llegar hasta el final.

Índice

1. Resumen

Este proyecto tiene la finalidad del diseño de un puente grúa automatizado de precisión, con la creación de un prototipo que simulará y probará el diseño. El puente grúa tiene que llevar a cabo la tarea de poder recoger un objeto en alguna posición predeterminada, que posteriormente tendrá que dejar en otra posición establecida anteriormente. El prototipo constará de tres ejes (X, Y y Z), podrá moverse en un espacio tridimensional, además tendrá una pinza que permitirá coger y soltar el objeto dependiendo de las ordenes de control.

El control del prototipo se hará con una placa Arduino Mega, una controladora de gran potencia con múltiples E/S, en la que se podrán conectar los múltiples motores, pantalla LCD y un teclado matricial. Esta placa controladora permitirá interactuar al usuario con el prototipo, a partir de la pantalla LCD y en el teclado el usuario podrá introducir las posiciones para recoger y dejar el objeto.

En la memoria del proyecto se explicará por separado el hardware utilizado, y el software diseñado, así como la construcción del prototipo.

2. Abstract

The purpose of this project is to design a precision automatic crane bridge, with the creation of a prototype that will simulate and test the design. The machine aim is achieve the task of catching an object placed in a predetermined position, and afterwards relasing the object on another position established beforehand.

The prototype will consist of three axes (X, Y and Z) and will be able to move in a three-dimensional space, as well as having a clamp which will allow the crane bridge to catch and drop the object depending on the central commands.

The prototype's control will be made with an Arduino board, a powerful controller with multiple E/S, that will permit to connect multiple motors, LCD screen and matrix keyboard. This controller board will empower the user to interact with the prototype by entering the catching and dropping positions on the LCD screen with the keyboard.

In the project report will be explained separately the hardware used, the software designed and the prototype construction.

3. Introducción

Los puentes grúa son maquinaria pesada capaces de poder mover toneladas de peso en un espacio de tres dimensiones. Normalmente esta maquinaria está posicionada en zonas industriales, con la necesidad de mover mucho peso sin tener en cuenta su precisión, ya que normalmente son de control manual, dirigido por un operario.

La aplicación del proyecto se basa en un sistema robotizado para manipular objetos de pequeño tamaño con precisión. Ya sea dentro de una habitación dónde el ser humano no pueda entrar por riesgo de contaminación u otros factores, o porque los elementos a manipular han de estar dentro de una cámara sellada o en una cadena de montaje que requiere alta precisión. El uso de maquinaria como esta puede facilitar la vida del ser humano. Gracias a la robótica, muchos trabajos peligrosos o de repetición los ha dejado de hacer el hombre, remplazados por máquinas y así no correr riesgos innecesarios e incrementar la productividad.

En este proyecto se utilizará una placa Arduino Mega 2560 para poder hacer el control del prototipo. Para mover los ejes del puente se usarán dos motores paso a paso, uno para mover el eje X y otro para el Y, habrá otro motor paso a paso el cual moverá el eje Z.

Mientras que en un puente grúa el eje Z es un sistema de poleas, que hace bajar una cadena, este hará bajar un eje fijo con una pinza acoplada, que con un motor paso a paso se podrá abrir y cerrar.

Para introducir las posiciones se utilizará un teclado flexible de 3 columnas por 4 filas. Aparte de usarlo para introducir posiciones, podemos darle otra función, cómo por ejemplo incluir una tecla de seguridad para confirmar posiciones. Para poder ver e interactuar con la placa, se utilizará una pantalla LCD de 20 dígitos en cada fila por 4 filas. Gracias a estos dos instrumentos, un usuario podrá controlar de una manera simple y eficaz los movimientos del puente grúa.

3.1. Objetivos del proyecto

El objetivo principal de este proyecto es el diseño electrónico de un sistema robótico de precisión con una placa de control Arduino. Crear un programa sencillo pero a la vez eficiente para un buen manejo del puente grúa. Construir un prototipo con la finalidad de comprobar el programa. Mediante otros instrumentos electrónicos y mecánicos, como el uso de varios motores paso a paso, o bien un teclado y una pantalla, se podrán elegir las posiciones deseadas, y mover los ejes del puente grúa hasta los puntos indicados.

Objetivos del proyecto:

1. **Diseño.** Diseñar un control de un sistema robótico capaz de moverse dentro de las tres dimensiones y capaz de coger pequeños objetos con precisión.
2. **Electrónica y motores paso a paso.** Buscar y seleccionar los instrumentos más indicados para realizar el prototipo, y con ellos automatizar el puente grúa con una interfaz directa usuario-controladora.
3. **Programación.** Crear desde cero un programa adecuado a nuestro proyecto, para poder realizar la automatización del puente grúa y de todos los instrumentos involucrados.
4. **Evaluación real.** Comprobar una vez construido el puente grúa que funciona según lo previsto.

3.2. Límites del proyecto

A nivel funcional se tienen que tener en cuenta algunas consideraciones. En primer lugar, el tamaño del prototipo está limitado, ya que no es posible hacer un prototipo a escala real, tanto por espacio como por el coste asociado. Este impedimento limita el tamaño de los componentes a usar, y por lo tanto eso significa motores menos potentes.

4. Contextualización

4.1. Automatización

En el ámbito de la automatización podemos encontrar diferentes instrumentos de control, para automatizar uno o varios sistemas electromecánicos a la vez. Desde un PLC (*Programable Logic Controller*) para controlar grandes cantidades de entradas y salidas de forma robusta, cómo podría ser una empresa de fabricación en cadena, controlando de manera automática todas y cada una de las máquinas que se utilicen en dicha empresa. También otro de los instrumentos de control más utilizados son las FPGA (*Field Programmable Gate Array*), estas tarjetas de control son algo más económicas que los PLC. Su uso es algo más limitado referente a la cantidad de actuadores que se pueden automatizar, pero a su favor tiene la rapidez y la manipulación de programas con un alto grado de complejidad. Fueron diseñadas a partir de los CPLD (*Complex Programmable Logic Device*), la arquitectura de los CPLD es más rígida y consiste en una o más sumas de productos programables. Mientras que, la arquitectura de las FPGA tienen una gran flexibilidad.

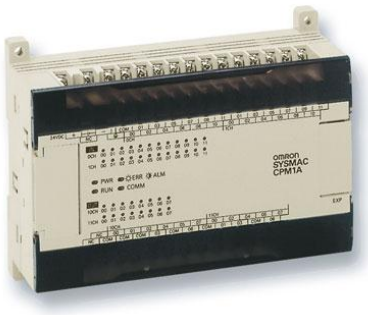


Ilustración 1 PLC OMRON CPM1A

Ilustración 2 FPGA Spartan BASYS2

La automatización es un ámbito que está en continuo desarrollo, siempre salen a la luz métodos o instrumentos para poder hacer programas más simples, pero a la vez igual o más eficientes.

Uno de los últimos instrumentos de control que ha tenido mucho éxito en el mercado, ha sido Arduino®. Gracias a su sencillo uso, su manejo es el óptimo para desarrollar prototipos o proyectos de baja magnitud.

4.2. Puente grúa

Un puente grúa, es un tipo de grúa que se utiliza en fábricas e industrias, para izar y desplazar cargas pesadas, permitiendo que se puedan movilizar piezas de gran porte en forma horizontal y vertical. Un puente grúa se compone de un par de rieles paralelos ubicados a gran altura sobre los laterales del edificio con un puente metálico (viga) desplazable que cubre el espacio entre ellas. El dispositivo de movimiento de la grúa, se desplaza junto con el puente sobre el cual se encuentra; a su vez la segunda parte se encuentra alojado sobre otro riel que le permite moverse para ubicarse en posiciones entre los dos rieles principales.

4.2.1. Aplicaciones

A diferencia de las grúas móviles o de construcción, los puentes-grúa son utilizados por lo general en fábricas, estando limitados a operar dentro de la nave industrial donde se encuentran instalados. El uso de este tipo de grúa se aplica en la industria del acero, para mover productos terminados, tal como, bobinas, caños y vigas, tanto para su almacenamiento, como para la carga a los transportes convenientes.

En la industria subsidiaria del cemento, para facilitar la fabricación de caños, postes, vigas, entre otros productos de gran peso y volumen. En la industria del automóvil y de maquinarias pesadas, se utilizan puentes grúa para el manejo de materias primas y en otros casos para el ensamblado de grandes piezas, en máquinas viales. [12]

Basándose en su diseño, este proyecto pretende obtener un equivalente en robustez pero de mayor precisión a una escala menor para la manipulación y posicionamiento preciso de pequeños objetos.

5. Instrumentación electrónica

5.1. Placa Arduino Mega 2560

Uno de los instrumentos de control más utilizados actualmente para prototipos o pequeños proyectos, son las placas Arduino. Hay mucha variedad de placas, y cada una de ellas hace funciones para determinadas ocasiones.

Se pueden encontrar desde de las placas más completas y competentes, como es la Arduino Mega 2560, hasta una placa más pequeña y reducida en tamaño, esta sería la Arduino Nano, una placa orientada a pequeños proyectos o donde por el espacio no es posible incluir placas más grandes. No por su tamaño deja de ser más mediocre, con un procesador ATMega168, consigue trabajar a 16MHz con un total de 14 pines digitales y 8 analógicos, se puede conseguir un gran rendimiento de esta placa.

La placa Arduino Mega 2560 es la que se utilizará en este proyecto, se ha elegido esta por el número de entradas/salidas digitales que tiene, sobre todo por la cantidad de pines PWM que tiene. Los motores NEMA-17 utilizan un pin PWM para poder hacer su control, mientras que el motor 28byj-48 con su placa controladora utilizan 3 pines PWM, por lo que se necesitan 8 para los 4 motores paso-a-paso (dos motores NEMA-17 y dos motores 28byj-48).

Esta placa trabaja a 16Mhz y con un voltaje de 5V. Sus capacidades son superiores al ATmega320 del Arduino UNO, aunque no tan superiores como las soluciones basadas en ARM. Este microcontrolador de 8 bits trabaja conjuntamente con una SRAM de 8KB, 4KB de EEPROM y 256KB de flash (8KB para el *bootloader*). El número de pines es de 54 pines digitales (15 de ellos PWM) y 16 pines analógicos. [1]

En la siguiente imagen, se pueden observar las diferentes partes a rasgos generales del Arduino Mega 2560.

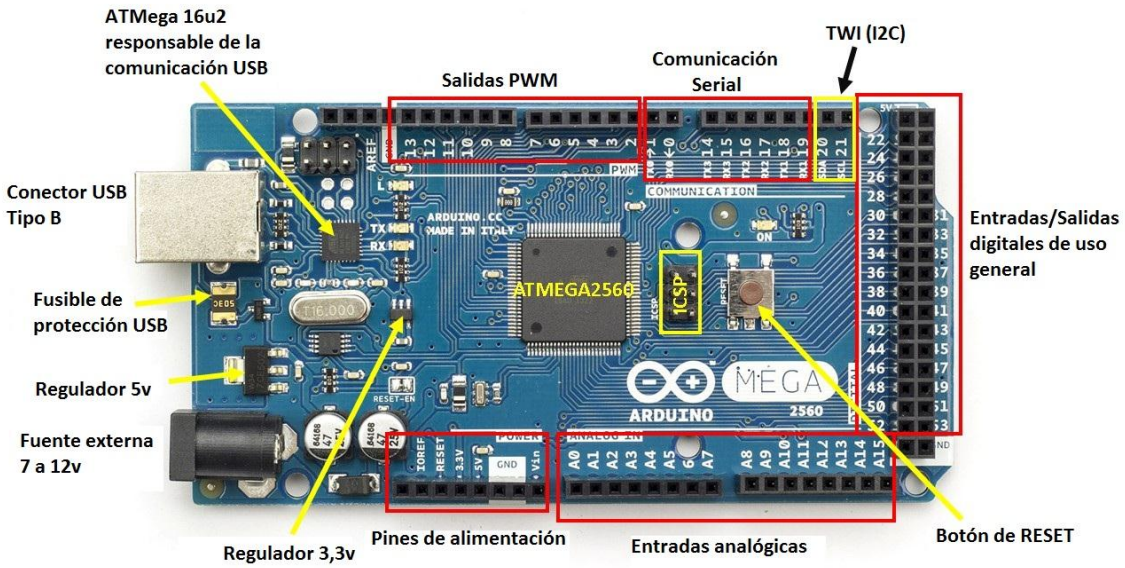


Ilustración 3 Arduino Mega 2560

En el esquema se pueden detectar todas las partes importantes de la placa, una de las partes vitales es la correcta alimentación al Arduino. Se alimenta con una fuente externa de entre 7 y 12V, por si se necesita coger tensión extra para un circuito del proyecto, o bien, alimentarlo vía puerto USB con una alimentación de 5v, el único inconveniente, es que la tensión proporcionada a la placa es más bien baja, i por lo tanto tendremos que utilizar las salidas de tensión con moderación.

Los pines analógicos sirven para el uso de sensores analógicos, cómo por ejemplo sensores de temperatura, sensores de presión, o algo más simple cómo un potenciómetro. El uso de los pines analógicos, son solo de entrada, por lo tanto, no abran actuadores conectados a estos pines.

Los pines digitales los podemos separar en dos grupos, las entradas/salidas digitales i las salidas PWM. Estos primeros son pines digitales comunes, se pueden de entradas, cómo el uso de sensores TODO/NADA, es decir, sensores con solo dos estados, “LOW” representado con el 0, o el “HIGH” representado con el uno. Realmente nos están dando dos tensiones, la tensión baja 0V y la tensión alta 5V, un claro ejemplo de estos sensores, es un fin de carrera. También se pueden usar de salidas digitales, los actuadores digitales se conectan a estos pines para recibir señales de tensión alta (5V) o baja (0V), uno de los actuadores más sencillos es el LED, con una tensión alta se iluminaría y con una tensión baja se apagaría. Otro ejemplo de actuador digital es la pantalla LCD, necesita conectarse a más de un pin digital de salida, ya que tiene muchas variables digitales cómo *enable*, el pin de escritura, entre otros.

PWM (*Pulse-width modulation*), modulación por ancho de pulso es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. En nuestro caso se envía una señal cuadrada, con un nivel alto de 5V y un nivel bajo de 0V pero esta modulado por una amplitud variable. El uso más significativo en este proyecto es el control de los motores paso a paso, ya que dependiendo de los pulsos se activan los diferentes estados del motor paso a paso.

La comunicación serial, es para establecer enlaces entre otros dispositivos, es decir, para poder conectar la placa Arduino con un ordenador se necesita un enlace por comunicación serial. El puerto de serie 1 está conectado a los pines 0 (RX) y 1 (TX), el puerto de serie 1 a los pines 19 (RX) y 18 (TX) el puerto de serie 2 a los pines 17 (RX) y 16 (TX), y el puerto serie 3 a los pines 15 (RX) y 14 (TX). Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). Uno de los instrumentos más comunes hoy en día para aplicaciones con Arduino, es el Bluetooth. También se puede usar el puerto serial para encender o apagar un dispositivo desde una página Web a través de Internet. [2]

5.2. Pantalla LCD HD44780

Para que un usuario se comunique con la placa y así poder interactuar de una manera cómoda y sencilla, se hará uso de una pantalla LCD complementándose con el Arduino. Se trata de una pantalla diseñada para ser controlada por micro-controladores, suele usarse en muchos proyectos y prototipos con placas de control Arduino, ya que su fácil implementación y su sencilla programación hacen de este instrumento de visualización, el perfecto para poder cumplir nuestros objetivos.

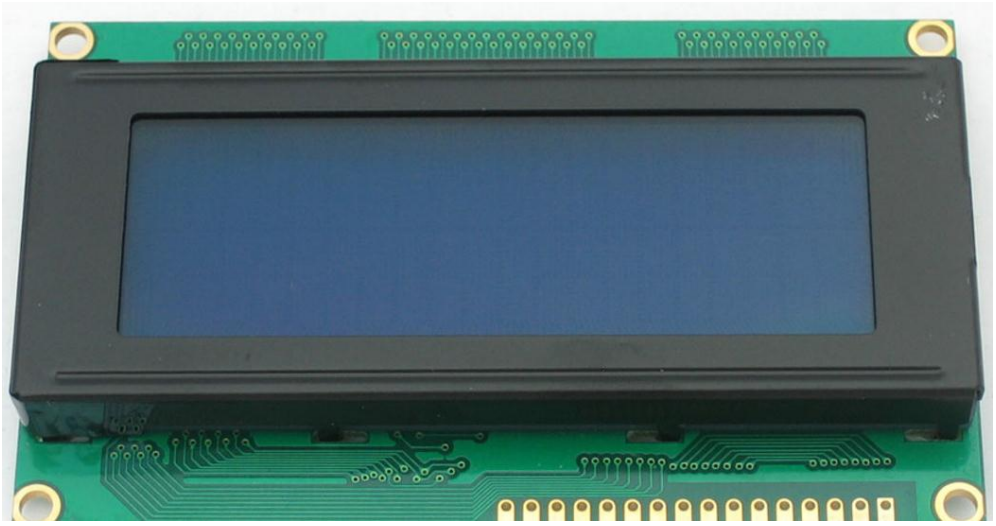


Ilustración 4 Pantalla LCD 20x4 HD44780

Esta pantalla LCD consta de 4 filas y tiene 20 dígitos por cada fila, admite la mayoría de letras, números y signos. Usa el código ASCII para poder interpretar la programación y así transformar un dígito ASCII en una letra, número o signo.

En la siguiente ilustración (ilustración 5) podemos ver la descripción de cada pin, los números de pin están ordenados de derecha a izquierda, representado en la Ilustración 4. Podemos observar también su símbolo y el tipo de conexión externa que tiene, junto a una breve descripción.

Interface pin description

Pin no.	Symbol	External connection	Function
1	V_{SS}	Power supply	Signal ground for LCM (GND)
2	V_{DD}		Power supply for logic (+5V) for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power Supply	Power supply for BKL (Anode)
16	LED-		Power supply for BKL (GND)

Ilustración 5 Descripción de los pines de la pantalla LCD

Descripción de pines de pantalla LCD:

1. Vss, pin 1, es la señal de GND; este pin se tiene que conectar a una tensión de nivel bajo (0V), es decir, en la línea de masa donde está conectada el Arduino.

2. Vdd, pin 2, es la señal de alimentación; la pantalla se alimenta con una tensión de +5V, esta señal la podemos extraer del pin de alimentación (+5V) que tiene el Arduino.

3. Vo, pin 3, es la señal de contraste en la iluminación de pantalla; para poder iluminar la pantalla de una forma eficiente y acorde a nuestras necesidades.

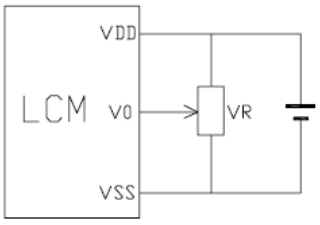


Ilustración 6 Esquema de conexión del contraste de la LCD

Tendremos que aplicar una resistencia variable (VR) de entre 10KΩ y 20KΩ entre las señales de VDD y VSS, tal y como se enseña en la siguiente ilustración (ilustración 6).

4. RS, pin 4, es la señal de registro; este pin lo conectamos a un pin digital de Arduino. Con una señal de estado bajo, significa escribir comando, si la señal por lo contrario es de estado alto, significa escribir datos.

5. R/W, pin 5, es la señal de leer y escribir; este pin lo conectamos a un pin digital de Arduino, trabajará como salida o entrada, dependiendo de la necesidad otorgará un 0 o 1 lógico; para escribir dará una señal de estado bajo (0 lógico), y para leer al contrario, una señal de estado alto (1 lógico). Este pin es opcional, ya que trabajando con la señal de registro tenemos suficiente, en caso de este proyecto se ha omitido y se ha conectado a la línea de 0v.

6. E, pin 6, es la señal para poder habilitar la lectura o escritura (*enable*); se conecta a un pin digital del Arduino. Cuando el Arduino tiene que escribir o leer, el *enable* se activa con flanco descendiente.

7. Pin 7-10, son DB0 a DB3 respectivamente; los pines DB0, DB1, DB2 y DB3 se dejan sin conexión, ya que estos pines si se trabaja con 4 bits no se utilizan, por lo que como trabajamos a 4 bits los dejaremos en el aire.

8. Pin 11-14, son DB4 a DB7 respectivamente; los pines DB4, DB5, DB6 y DB7 se conectan a pines digitales del Arduino. A estos pines se le darán señales de estado bajo/alto dependiendo del código de programación.

9. LED+, pin 15, es el ánodo del LED encargado de la iluminación, aplicaremos una tensión de +5v que podemos extraer de los pines de alimentación de Arduino.

10. LED-, pin 16, es por el contrario el cátodo del LED, conectaremos este pin a la línea de GND de los pines de alimentación de Arduino, dándole una tensión de 0v. [3] [4]

5.3. Teclado numérico flexible matricial 3x4

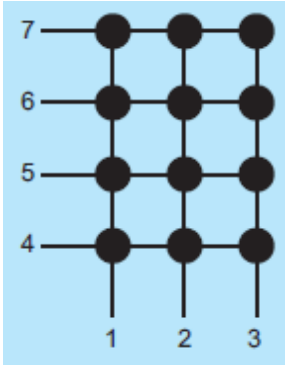
La comunicación del usuario con la placa Arduino es gracias a un instrumento de visualización, cómo es la pantalla LCD, y un elemento de escritura, en el caso de este proyecto, un teclado numérico matricial de 3 columnas por 4 filas.

En la siguiente ilustración (ilustración 7) podemos observar la composición del teclado, por defecto el teclado numérico estándar tiene los números del 0 al 9, además de dos botones adicionales que son los signos de asterisco "*" y de almohadilla "#".

En este proyecto los botones numéricos se usarán para indicar la posición de recoger el objeto y la posición de dejarlo.

Tendremos 9 posiciones, la posición inicial estará representado con el botón "0", el botón "1" representará la posición 1 y así respectivamente hasta el botón "7". Los botones "8", "9", "*", no se usarán para ningún comando, mientras que el botón "#" servirá para confirmar las posiciones que se hayan indicado anteriormente.

Ilustración 7 Teclado numérico flexible matricial de 3x4



En la ilustración 8 se puede observar cómo están ordenados los pines del teclado matricial, la distribución de los pines se puede ver en la ilustración 7, donde el primer pin de la izquierda es el pin 1 y el último de la derecha es el pin 7.

Estos se conectan a pines digitales de Arduino para poder usarlo con su respectiva programación. [4]

Ilustración 8 Esquema matricial de la organización de los pines

5.4. Motor driver L298N

En este proyecto se utilizaran motores paso a paso Nema 17, los cuales necesitan la ayuda de *drivers* de potencia. Estos les proporcionan la suficiente alimentación y potencia que necesitan para tener un óptimo uso de los motores.

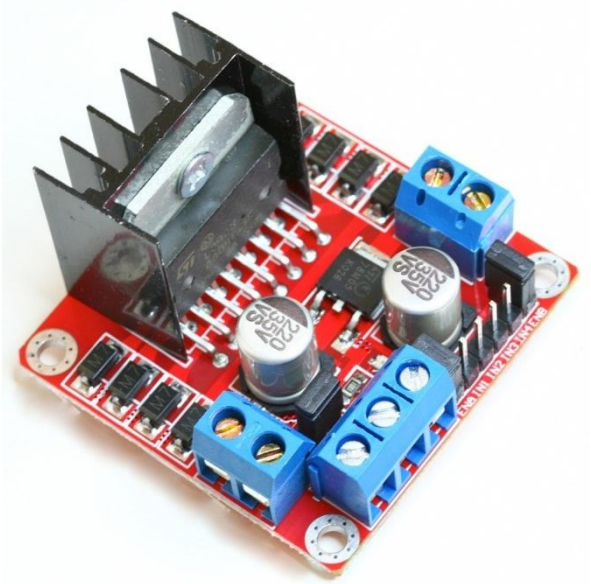


Ilustración 9 Driver de potencia L298N Dual motor

El uso de estos *drivers* de potencia es totalmente necesario, ya que sin ellos los motores no podrían alimentarse correctamente.

El L298N tiene la particularidad de tener dos funciones, cómo fuente de potencia para dos motores DC o bien para el suministro de potencia de un motor paso a paso.

En este caso se utilizará como *driver* de potencia para el uso de un motor paso a paso. Se necesitaran dos *drivers* L298N, ya que tenemos dos motores Nema17 que requieren de este *driver* para funcionar correctamente.

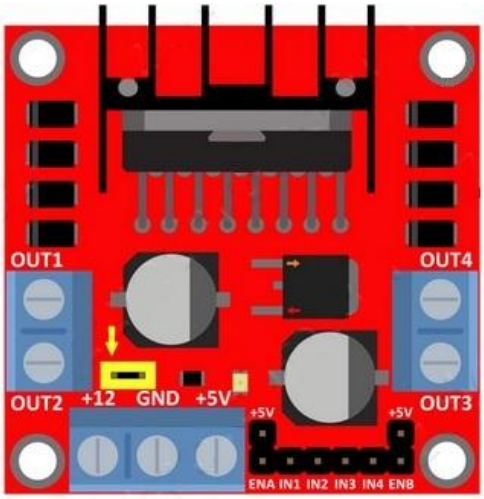


Ilustración 10 Esquema del driver L298N

En este esquema (Ilustración 10) se puede observar a grandes rasgos, los componentes del L298N más importantes.

Descripción de las partes importantes del L298N:

1. Out1, Out2, Out3 y Out4; son las salidas directas al motor, los motores Nema17 tienen 4 cables, los cuales se tienen que conectar aquí por orden. Si no se conectasen en orden, el funcionamiento del motor paso a paso no sería el adecuado.

2. IN1, IN2, IN3 y IN4; son los pines de entrada, se conectan a pines digitales de Arduino para poder controlar el motor.

3. ENA y ENB; son los *enable*, estos pines los dejaremos conectados con un *jumper* con los pines superiores de +5V. Los ENA y ENB no se usaran en este proyecto ya que su función es para el control de motores DC.

4. La alimentación del *driver* es variable, ya que tiene un regulador de tensión de 5v. Cuando el *jumper* de selección de 5V se encuentra activo, el módulo permite una alimentación de entre 6V a 12V DC. Como el regulador se encuentra activo, el pin marcado como +5V tendrá un voltaje de 5V DC. Este voltaje se puede usar para alimentar la parte de control del *driver* ya sea un micro-controlador o un Arduino, pero recomendamos que el consumo no sea mayor a 500 mA.
   Cuando el jumper de selección de 5V se encuentra inactivo, el *driver* permite una alimentación de entre 12V a 35V DC. Como el regulador no está funcionando, tendremos que conectar el pin de +5V a una tensión de 5V para alimentar la parte lógica del L298N. [7]

En este caso se utilizará una fuente de 12V manteniendo el *jumper* activo, y así alimentar la parte de control del *driver*.

5.5. Motor driver ULN2003

El uso de diferentes motores paso a paso, implica utilizar *drivers* convenientes para cada motor. En particular el *driver* ULN2003, se utilizará para potenciar la salida del motor paso a paso 28BYJ-48. [6]

Ilustración 11 Driver de potencia ULN2003

El driver ULN2003 es un circuito integrado, acoplado de una sencilla forma en una placa de conexión.

Donde se conectará una parte a la placa controladora, en este caso es un Arduino, y otra al motor paso a paso con un conector muy dinámico e intuitivo. En la siguiente ilustración (ilustración 12) se puede observar cada una de las partes más importantes de este *driver*.

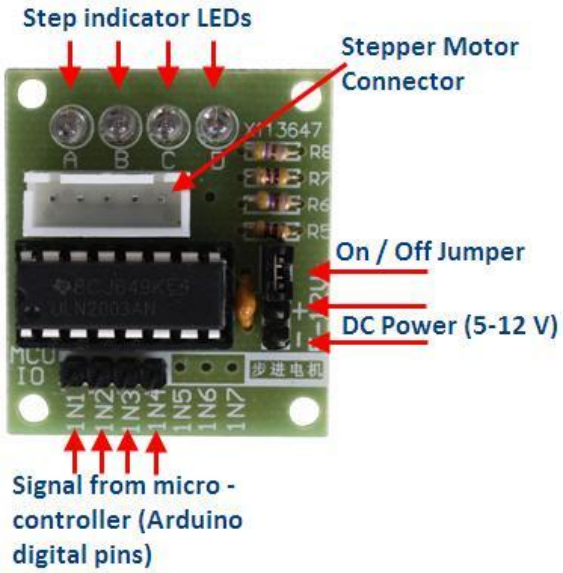


Ilustración 12 Esquema driver de potencia ULN2003

Esta placa de potencia es bastante más sencilla que la vista anteriormente (L298N), Hay 5 partes a destacar:

-Señal de control, esta es la entrada del driver, van conectados a pines digitales del Arduino, para poder hacer el control del motor.

-Señal de salida, es el conector del motor paso a paso, para poder hacer una conexión con el motor y así proporcionarle suficiente potencia.

- Alimentación, este *driver* se alimenta entre 5-12V, para poder utilizarlo a su mejor rendimiento, debería conectarse a una fuente de 9v, ya que conseguiremos que el motor funcione con un buen rendimiento, y con la seguridad de no quemar el componente por un pequeño pico de tensión.

- *Jumper,* es el puente que hay al lado de la alimentación, con él conectado conseguiremos mantener la potencia del motor paso a paso.

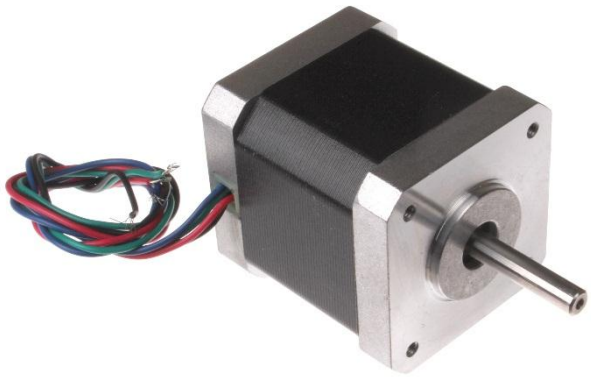
-*Leds,* son los indicadores del motor, es decir, se iluminarán los *leds* dependiendo de los pasos que esté efectuando el motor, si está haciendo mover el *stepper* AB, se iluminarán los *leds* correlativos A y B.

6. Instrumentación electro-mecánica

6.1. Motor paso a paso Nema17

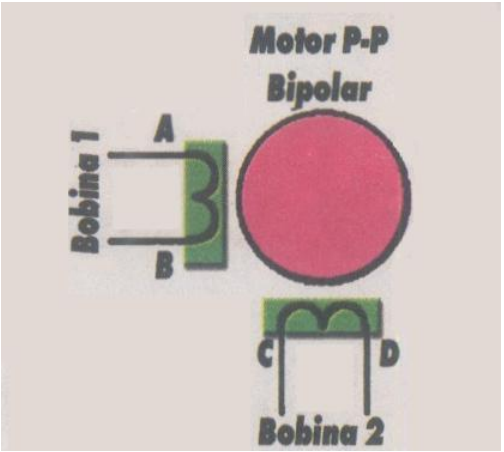
El movimiento de los ejes del puente grúa lo proporcionarán los motores paso a paso, en el caso del motor Nema17 se usará para mover el eje "Y" y otro motor Nema17 para el eje "X".

Estos motores están constituidos normalmente por un rotor sobre el que van aplicados distintos imanes permanentes y por un cierto número de bobinas excitadoras en su estator. Las bobinas son parte del estator y el rotor es un imán permanente. Toda la conmutación (o excitación de las bobinas) deber ser externamente manejada por un controlador.



El Nema17 es un motor paso a paso bipolar de 4 cables, por lo que tendremos 4 fases para accionar los diferentes estados y así hacer que rote el eje del motor.

Ilustración 13 Motor paso a paso Nema17



Para hacer que gire el motor debidamente, se requiere un cambio de dirección del flujo de corriente a través de las bobinas, en la secuencia apropiada para realizar un movimiento.

Ilustración 14 Esquema de un motor paso a paso bipolar de 4 cables

En la ilustración 14 podemos observar el esquema de las bobinas conectadas cada una a dos cables diferentes (A-B-C-D), que accionándolas de manera apropiada y siguiendo una secuencia determinada podremos hacer mover el motor hacia un lado o hacia otro.

En la siguiente Ilustración (ilustración 15) se puede observar la tabla con la secuencia necesaria para controlar motores paso a paso del tipo bipolares.

PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	ON	ON	OFF	OFF	
2	OFF	ON	ON	OFF	
3	OFF	OFF	ON	ON	
4	ON	OFF	OFF	ON	

Ilustración 15 Tabla de secuencia de fases para un motor paso a paso bipolar

Viendo esta tabla podemos saber que fases se tienen que activar para poder girar en sentido horario o anti-horario.

Para girar en sentido horario, la secuencia es: AD-DC-CB-BA

Para el sentido anti-horario, la secuencia es: AB-BC-CD-DA

Tiene un ángulo de paso de 1.8º, es decir, cada paso el motor gira 1,8º y para poder hacer una vuelta completa (360º), dividimos 360/1,8 y da el resultado de 200 pasos para poder hacer una vuelta. Tenemos que tener en cuenta este dato, ya que es muy importante a la hora de programar, si no se sabe cómo trabaja el motor, no se podrá programar adecuadamente y por lo tanto no conseguiremos que funcione correctamente.

Este motor es el modelo 42BYGHW811, el cual trabaja a 2,5A de corriente nominal, y 3,1V de tensión nominal. La corriente se la proporciona su respectivo driver de potencia, dándole una tensión al driver de 5V, tenemos suficiente para alimentar al motor. Pudiendo proporcionar esta corriente con su respectivo voltaje, este motor es capaz de cargar con 4,8 kg·cm de par. [8]

6.2. Motor paso a paso 28BYJ-48

Este motor es mucho más pequeño, manejable y con mucho menos torque que el anterior, pero se ha elegido para hacer una serie de tareas donde no necesita levantar mucho peso, y donde no hay espacio para usar un Nema 17. Ya que requerimos de poco espacio, el motor 28BYJ-48 es el ideal. Utilizaremos dos motores 28BYJ-48, uno para el eje Z, y otro para el mecanizado de la pinza.

Ilustración 16 Motor paso a paso 28BYJ-48

Cómo se puede observar en la Ilustración 18, el motor 28BYJ-48 es de 5 cables, comparado al Nema 17, este tiene un cable más. Sigue siendo un motor de 4 fases, pero este cable adicional es el COMMON directo que tiene a los bobinados.

Es un motor que tiene una relación de reducción de 1:64 por lo que tiene un ángulo de paso de 5,625º, que se puede obtiene dividiendo el recorrido por 64 (360º/64). Este motor necesita dar un total de 2025 pasos para poder dar una vuelta completa del eje principal. Para saber los pasos necesarios para dar una vuelta completa, multiplicamos el ángulo de paso $5{,}625^{o} \cdot 360^{o}$ y nos da un total de 2025 pasos necesarios para dar una vuelta.

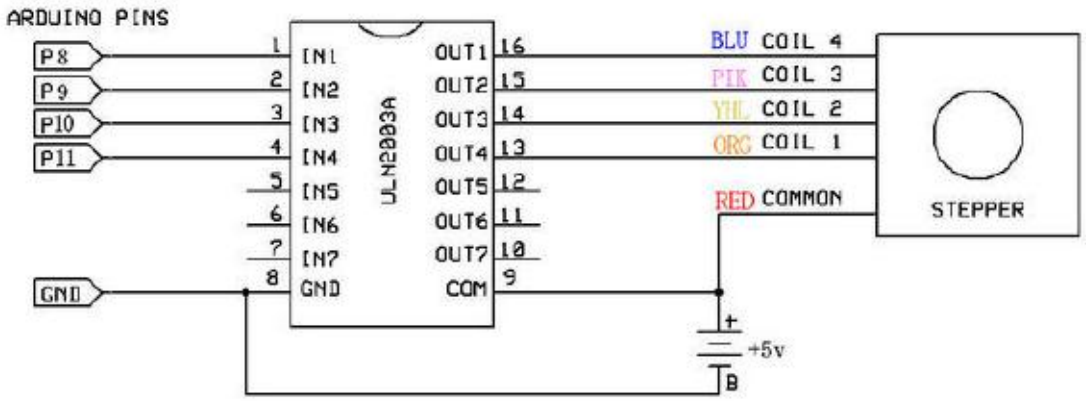


Ilustración 17 Esquema motor paso a paso 28BYJ-48 con driver ULN2003

En la Ilustración 17 se pueden observar las conexiones entre el motor i el *driver*, la conexión que llama más la atención es el cable RED, este a diferencia del Nema 17 que no lo tiene es un cable común entre los dos bobinados. En la Ilustración 20 se puede ver que es un punto de tensión común entre las dos bobinas del motor.

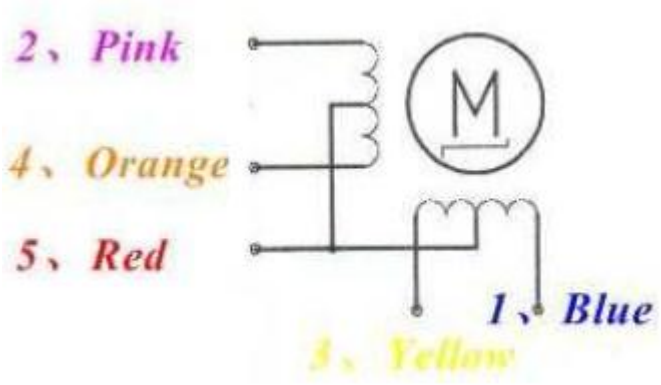


Ilustración 18 Esquema bobinado motor paso a paso 28BYJ-48

Este motor tiene dos variaciones el de tensión de 5V o de 12V, en este caso se utilizarán de 5V, por lo que el driver ULN2003 se alimentará a 5V. El par de este motor es mayor a 34,3mN·m, cómo se ha mencionado antes el par es más pequeño que el del Nema 17. [4]

7. Arduino

7.1. Introducción

Cómo se ha nombrado anteriormente, el control del prototipo se hará con una placa Arduino Mega 2560. Arduino es una empresa fundada por Massimo Banzi, un estudiante de instituto; en un principio comenzó siendo un proyecto para el aprendizaje de otros estudiantes. Después de más trabajo aplicado en este prototipo inicial, consiguieron evolucionar y desarrollarse. Hoy en día es una de las placas de control más vendidas, ya sea por su precio económico o bien por la amplia gama de tipos de placa.

Ilustración 19 Logo compañía Arduino

El lenguaje de programación de Arduino es propio, basado en el lenguaje de programación de alto nivel *Processing* que es similar a C++. *Processing* es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Al estar basado en Java, puede heredar todas sus funcionalidades, convirtiéndose en una herramienta poderosa a la hora de afrontar proyectos complejos. [9]

7.2. Programación

La explicación de la programación del prototipo se hará seccionado, y así dejar claro cada código de programación. El código completo sin cortes se podrá encontrar en los anexos. Hay tres partes a explicar: Entrada de variables, *setup()* y *loop().*

7.2.1. Entrada de variables

```
/* [Trabajo final de Grado]

"Puente grúa automatizado"
EPSEM UPC
Ingeniería Electrónica Industrial y Automática
Dani Andrade García

*/
```

En la cabecera ponemos el título, nombre y entidad del proyecto. Siempre que queramos poner explicaciones, nombres u otros datos los pondremos entre /* y */, de esta forma podemos hacer más de una línea de explicación sin temor a que nos lo detecte cómo código.

Si lo que necesitamos es una simple explicación de una línea o un apunte de información pequeño, usaremos // antes de hacer dicha explicación.

Para asegurarse de que la nota de información no nos la cogerá cómo código, nos fijaremos en el color del texto, si es gris, significa que está detectando la información cómo texto y no cómo código, si por el contrario es negro, significará que lo detecta cómo parte del código.

7.2.1.1. Librerías

```
#include <Keypad.h>
#include <Stepper.h>
#include <LiquidCrystal.h>
```

La entrada de librerías son vitales en muchos proyectos, ya que sin ellas no se podrían usar muchos de los instrumentos, tanto sensores cómo actuadores.

Las librerías son archivos comprimidos y pre-programados dónde se alojan las funciones y variables necesarias para controlar el componente o instrumento indicado. Para introducir una librería dentro de un código se hace en la entrada de variables, la mayoría de librerías están incluidas en el programa Arduino, pero otras se tienen que descargar desde la página oficial, cómo es el caso del *<Keypad.h>.*

Con la función [#include] seguido <Componente.h> dónde componente es el elemento de la librería, cómo por ejemplo `#include <Keypad.h>`, introducimos la librería del teclado. En este proyecto se han necesitado tres librerías adicionales, la del teclado, otra para la pantalla LCD y la librería para el control de los motores paso a paso.

7.2.1.2. Pantalla LCD

```
LiquidCrystal lcd(22, 24, 26, 28, 30, 32);
```

La pantalla LCD necesita conectarse con 6 pines digitales al Arduino, el pin 4, o RS, el pin 6, o E, y los pines del 7-10, es decir, DB4-DB7. Para que la placa de control reconozca estas E/S cómo parte de la pantalla LCD, se necesita de una función de introducción de variables y en ella se introducen los pines en orden.

La función es la siguiente: LiquidCrystal lcd(RS, E, DB4, DB5, DB6, DB7); de esta forma conseguiremos interpretar los pines cómo E/S de la pantalla digital.

Siempre después de introducir una función o variable acabaremos con un “;” para que el Arduino nos la reconozca como tal.

7.2.1.3. Teclado matricial (Keypad)

```
byte rowPins[ROWS] = {31,33,35,37};
byte colsPins[COLS] = {39,41,43};
```

Para establecer los pines cómo entradas del Arduino para el teclado matricial, se necesita la función `byte`, con ella declararemos una variable para las filas (rowPins) y otra para las columnas (colsPins). Seguidamente introduciremos una variable constante que serán el número de filas (ROWS) y el número de columnas (COLS), y por último van los pines de las filas y columnas respectivamente.

```
const byte ROWS = 4; // Cuatro filas
const byte COLS = 3; // Tres columnas
```

Se necesita crear dos variables constantes, determinando cada una de ellas el número de filas y columnas respectivamente. Para ello, se usará la función `const byte` seguidamente se escribe el nombre de cada variable, en este caso será (ROWS) para las filas, hay un total de 4 filas, por lo que introduciremos un símbolo de "=" con un 4 para establecer dicha variable y (COLS) para columnas con un total de 3 columnas.

```
char tecla;
char tecla2;
char tecla3;
char pos1;
char pos2;
```

Para este programa se necesitarán una serie de variables que se utilizarán en el *loop()*, con la función `char` declaramos una variable como un carácter, capaz de almacenar tanto números, letras o símbolos.

```
/*  {     '1'  '2'  '3'    } Estas son las nueve posiciones numéricas
    {     '4'  '5'  '6'    } posibles, la posición 0 es el estado
    {     '0'  '7'  '8'    } inicial.
Estas posiciones no coinciden con el teclado totalmente, ya que
corresponden a los movimientos que hará el puente grúa.
*/
```

```
// Matriz que dibuja el teclado

char numberKeys[ROWS][COLS] ={

  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}

};
```

Para que Arduino reconozca cada número y símbolo una vez se pulse cualquier botón del teclado, se tiene que dibujar la matriz de números y símbolos dentro de una función. Una vez se pulse el botón, esta cogerá la variable dentro de la matriz dibujada y la guardará como un carácter. De esta manera conseguiremos tratar las variables cómo caracteres, y así poder utilizarlas sin ningún problema en la pantalla LCD.

```
//inicializar una instancia de la clase NewKeypad

Keypad customKeypad=Keypad(makeKeymap(numberKeys),rowPins,colsPins,ROWS,COLS);
```

Una vez establecido todas las variables del teclado, tenemos que iniciar una instancia, la cual creará dentro del micro-controlador el diseño del teclado digitalizado. Para cuando necesitemos utilizarlo, solo escribiendo la variable global podamos hacer un control adecuado.

En el caso de este proyecto se ha escogido la variable *customKeypad*, donde siguiendo la función escrita anteriormente, se puede observar todas las variables introducidas en orden, que se han escrito en las anteriores funciones.

7.2.1.4. Motor paso a paso (Stepper)

Los motores paso a paso son un elemento fácil de usar y programar, y debido a su gran versatilidad, se pueden ajustar perfectamente a las funciones deseadas.

Por lo que respecta a la entrada de variables de los motores paso a paso, hay que tener en cuenta la disposición de los pines, las constantes de pasos por revolución, que determinan cuantos pasos son necesarios para hacer una vuelta, y las constantes de pasos que van a realizar los motores.

```
//Si los pasos son positivos va en sentido anti horario, si son
negativos sentido horario

const int pasos1 = 165;
const int pasos2 = 330;
const int pasos3 = 230;
const int pasos4 = 460;
const int pasos5 = 4050;
const int pasos6 = 675;
```

Se introducen 6 constantes para los pasos a realizar, cada una de estas constantes servirá para otorgar un movimiento a los motores.

Tabla 1 Tabla de accionamientos de los pasos

Pasos 1	Accionará la mitad del movimiento en el motor del eje Y. (Consiguiendo mantenerse en las posiciones 4, 5 y 6)
Pasos 2	Accionará el movimiento total en el motor del eje Y. (Consiguiendo desplazarse de las posiciones 0, 7 y 8 a la 1, 2 y 3)
Pasos 3	Accionará la mitad del movimiento en el motor del eje X. (Consiguiendo mantenerse en las posiciones 2, 5 y 7)
Pasos 4	Accionará el movimiento total en el motor del eje X. (Consiguiendo desplazarse de las posiciones 1, 4 y 0 a la 3, 6 y 8)
Pasos 5	Accionará el movimiento total en el motor del eje Z. (Moviendo el eje desde su inicio hasta su final de recorrido)
Pasos 6	Accionará el movimiento total en el motor de la pinza. (Moviendo el mecanismo de la pinza para cerrarla y abrirla)

```
// Cada 2025 pasos son una vuelta completa del eje del motor 3 y 4
// Cada 200 pasos del motor 1 y 2


const int pasosxrevolucion = 200;
// pasos por revolución para el motor Nema 17
const int pasosxrevolucion2 = 2025;
// pasos por revolución para el motor 28byj-48
```

Se necesita otro tipo de constante para que el motor paso a paso funcione correctamente, esta es para indicar cuantos pasos se necesitan para realizar una vuelta completa del eje del motor. Al tener dos tipos motores diferentes, se necesitan dos constantes. Para ello se necesita haber calculado anteriormente este dato, en el caso del Nema 17 se necesitan 200 pasos, mientras que para el 28byj-48 se necesitan 2025.

```
Stepper myStepper(pasosxrevolucion, 38,3,4,2);
//Motor eje X [3,4,2 son PWM] [por orden de 1N1,1N3,1N2,1N4]
Stepper myStepper2(pasosxrevolucion, 36,6,7,5);
//Motor eje Y [6,7,5 son PWM]
Stepper myStepper3(pasosxrevolucion2, 34,9,10,8);
//Motor eje Z [9,10,8 son PWM]
Stepper myStepper4(pasosxrevolucion2, 40,12,13,11);
//Motor pinza [12,13,11 son PWM]

//cuando (pasos) es negativo en el eje horizontal (eje x) es que el
motor va a la derecha de lo contrario izquierda

//cuando (pasos) es negativo en el eje horizontal (eje y) es que el
motor va hacia arriba de lo contrario abajo

//cuando (pasos) es negativo en el eje vertical (eje Z) es que el
motor va hacia abajo de lo contrario arriba

//cuando (pasos) es negativo la pinza se abre, de lo contrario se
cierra
```

Por último, se necesitan identificar cada motor con sus pasos por revolución y con sus respectivas variables. Los motores *myStepper* y *myStepper2* son los motores (Nema 17) del eje X y eje Y respectivamente, estos irán con la constante de 200 pasos por revolución, mientras que los otros dos motores (28byj-48) son del eje Z y de la pinza. La introducción de los pines es muy importante, ya que si no se escriben de una manera ordenada, es muy posible que el motor no funcione. El orden adecuado es de 1N1, 1N3, 1N2, 1N4, sin este orden el control de los bobinados no es el correcto y por lo tanto no funcionaría como se precisa.

7.2.2. Setup

La función de configuración (*setup*) debe contener la declaración de las variables. Es la primera función a ejecutar en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar *pinMode* (modo de trabajo de las E/S), configuración de la comunicación en serie y otras.

La función *setup()* se invoca una sola vez cuando el programa empieza. Se utiliza para inicializar los modos de trabajo de los pines, o el puerto serie. Debe ser incluido en un programa aunque no haya declaración que ejecutar. Así mismo se puede utilizar para establecer el estado inicial de las salidas de la placa.

Las llaves ({ }) sirven para definir el principio y el final de un bloque de instrucciones. Se utilizan para los bloques de programación *setup()*, *loop(), if*.., etc. [10]

7.2.2.1. Inicialización LCD

```
void setup()
{

  //Inicio del programa
  //La pantalla LCD mostrará durante 5 segundos el título del proyecto

  lcd.begin(20, 4);
  lcd.setCursor(5,0);
  lcd.write("EPSEM UPC");
  lcd.setCursor(4,2);
  lcd.write("PUENTE GRUA");
  lcd.setCursor(4,3);
  lcd.write("AUTOMATIZADO");
  delay (5000);
```

Cuando se inicie el programa, se mostrará en la pantalla LCD el título del proyecto. Para ello es necesario usar los comandos `lcd.begin`, `lcd.setCursor` y `lcd.write`. El primero, *lcd.begin* tiene la función de iniciar la pantalla, insertando el número de dígitos que tiene por fila y el total de filas, en este caso es de 20 dígitos por 4 filas. El comando *lcd.setCursor* es el encargado de indicar donde comenzar a escribir, es decir, aplicar un cursor para poder escribir.

A continuación se introduce el número del dígito donde debe empezar y después el número de fila que pertenece, por ejemplo, `lcd.setCursor(5,0);` significa que empieza en la posición 5 de la primera fila (fila 0).

Por último tenemos el comando *lcd.write*, este como su nombre indica sirve para escribir. La pantalla mostrará lo que escribamos en el código, pero para que se muestree, se necesita escribir el código entre estos símbolos: ("*texto*").

Una vez escrito el texto, es necesario que se mantenga durante un espacio de tiempo definido, para que el usuario que esté usando el programa pueda leerlo sin dificultades. Con un comando de *delay* (espera) detrás de los comandos de escritura de la pantalla LCD, decidimos cuanto tiempo hay que mantener el texto en la pantalla, el tiempo se expresa en milisegundos, por lo que se deja 5000ms, un total de 5 segundos. Este texto solo es necesario que se lea al inicio del programa, por eso mismo se escribe el código dentro de la función *setup()*.

7.2.2.2. Configuración de velocidades

```
myStepper.setSpeed(60); //Determina la velocidad del motor1
myStepper2.setSpeed(60); //Determina la velocidad del motor2
myStepper3.setSpeed(14); //Determina la velocidad del motor3
myStepper4.setSpeed(14); //Determina la velocidad del motor4
```

Cómo se ha comentado anteriormente, en la función *setup()* se introducen datos iniciales que determinaran un funcionamiento en la función *loop()*. En el caso de los motores paso a paso se tienen que determinar las velocidades cómo configuración previa a su uso en la función *loop()*. Para ello se utiliza el comando `myStepper.setSpeed(velocidad);` donde *myStepper* es el nombre del motor a configurar, y entre paréntesis se introduce la velocidad deseada para cada motor. Es muy importante saber qué velocidad máxima tiene cada tipo de motor, por ejemplo, el motor Nema 17 puede llegar a una velocidad máxima de $60min^{-1}$, mientras que el 28byj-48 se limita a $14min^{-1}$. Si se introduce una velocidad por encima de su máxima, el motor dejará de funcionar no asimilando la velocidad establecida. [8]

7.2.2.3. Función anti-rebote

Unos de los mayores problemas que nos encontramos al trabajar con micro-controladores o Arduino es lo sensibles que son las entradas digitales a ruidos electromagnéticos, si las distancia a la que se encuentra la entrada es corta podemos utilizar una simple resistencia *pull-up* o *pull-down* y con eso se soluciona el problema, pero claro cuando las distancia son largas el cable tiende hacer de efecto antena "empapándose" de todo el ruido ambiente y haciendo que este llegue al Arduino.

Otro problema que nos podemos encontrar al contar una señal es que esta produzca rebotes, es muy normal usar pulsadores o interruptores en alguna de las entradas del Arduino. Estos pulsadores no hacen una conexión perfecta e instantánea como podemos pensar: un pulsador está compuesto de dos partes de metal que entran en contacto (choca una con la otra) al accionarlo.

Este choque genera unos rebotes que suceden tan rápido que son imperceptibles para nosotros, Sin embargo, no lo son para el PIC o Arduino que trabaja a esas velocidades. [11]

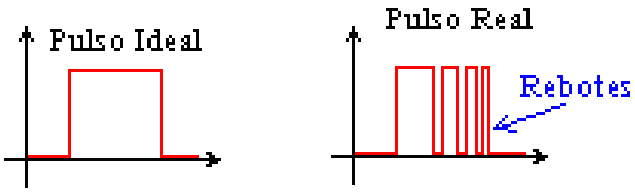


Ilustración 20 Comparativa de pulso real/ideal

```
customKeypad.setDebounceTime(50);
  //Función anti-rebote para los botones del teclado
```

Hay varias maneras de crear una función anti-rebotes, una de ellas y la más rápida y eficiente para los botones de un *keypad* es la propia función: `CustomKeypad.setDebounceTime(tiempo);`

Donde *customKeypad* es el nombre del teclado matricial, y en tiempo introducimos el pulso de espera en milisegundos.

En el caso, se ha introducido 50ms, es una demora de tiempo muy corta, pero realmente para un microcontrolador es suficiente para determinar si el pulso ha sido un pequeño rebote por culpa de ruido o si ha sido un botón del teclado que ha sido pulsado, esta función evitará muchos problemas de ruido, y de malas conexiones a los pines digitales.

```
  lcd.clear();
}
```

Por último en la función *setup(),* borraremos el texto de la pantalla LCD, y así dejar paso a la función *loop()* una pantalla limpia para poder escribir nuevos textos.

Para realizar dicha acción se necesita un comando muy sencillo, `lcd.clear();` cómo su nombre indica, dejará limpia la pantalla LCD para posteriores entradas de texto.

7.2.3. Loop

Después de llamar a *setup()*, la función *loop()* hace precisamente lo que sugiere su nombre, se ejecuta de forma cíclica, lo que posibilita que el programa esté respondiendo continuamente ante los eventos que se produzcan en la placa.

La función *loop()* contiene el código que se ejecutara continuamente (lectura de entradas, activación de salidas, etc) Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo. [10]

7.2.3.1. Inicio de programa

El programa principal, es decir, el programa que hará funcionar el prototipo a partir de señales de salida y entrada, está situado aquí en la función *loop().* Pero primero se necesita configurar la comunicación usuario-controladora a partir de la pantalla LCD y el teclado matricial. Si el usuario no puede interactuar debidamente con la placa controladora, de nada sirve la programación de los motores y por tanto del uso del prototipo.

```
void loop()
{

  lcd.display();
  //Enciende el display

  lcd.setCursor(0,0);
  lcd.write("Dónde quieres");
  lcd.setCursor(0,1);
  lcd.write("cogerlo?");
  lcd.setCursor(0,2);
  lcd.write("Dónde quieres");
  lcd.setCursor(0,3);
  lcd.write("dejarlo?");
```

Para poder interactuar con la placa controladora, es elemental que esta pregunte al usuario las condiciones o variables necesarias para hacer funcionar el prototipo. Para ello se ha programado una serie de preguntas que se visualizarán en la pantalla LCD, y así de una forma clara y sencilla el usuario podrá comunicarse con la controladora.

Con el comando `lcd.display();` encenderemos la pantalla, por si alguna vez el *display* se apaga por culpa de algún ruido y mala señal, el *display* se volverá a encender en cada bucle.

Cómo se explicó anteriormente en el apartado *Setup*, se programa un texto para que lo muestre mediante sus respectivos comandos, en este caso no solo pretende mostrar un texto, sino interactuar con el usuario. Con las preguntas "Dónde quieres cogerlo?" y "Dónde quieres dejarlo?" se hace decidir al usuario donde querrá coger el objeto y dónde dejarlo, a partir de aquí con el teclado numérico introducirá las respectivas posiciones en orden.

El usuario deberá indicar la primera posición y seguidamente la segunda, para que el programa no salte a otra instrucción y se quede bloqueado hasta recibir todas las ordenes, es necesario un comando esencial en este programa, este es *waitForKey()*.

Este comando permite hacer esperar al programa hasta que una tecla no ha sido pulsada. Para realizar la función correctamente, se introduce el nombre de la variable de la tecla, en este caso la variable es '*tecla*'. También es necesario introducir el nombre que se le ha otorgado al teclado, '*customKeypad*'.

```
char tecla = customKeypad.waitForKey();
// Espera hasta pulsar la tecla
// Guarda valor pulsado en la variable tecla
```

Una vez pulsada la tecla, esta se guardará en la variable que se haya establecido, se tiene que tener en cuenta, que la variable guardada esta codificada en ASCII, por lo que si se quiere utilizar dicha variable habrá dos opciones, o se trata en código decimal o bien se pasa el código a otro antes de utilizar la variable.

En este caso, se guardará la variable directamente en decimal, ya que la pantalla LCD pasa el código decimal a ASCII. Por ejemplo, si la tecla presionada es el '1' (ASCII) la controladora lo asimilará y le otorgará a la variable '*tecla*' un '49' (DEC). Cada una de las teclas tendrá un código decimal, para poder saber cuál es, se necesita la tabla de código ASCII. Esta se puede encontrar en los anexos.

El uso de condicionales es muy común en programas de Arduino, son instrucciones muy básicas pero a la vez muy útiles. Esta instrucción se utiliza junto a una comparativa, y así formular una "pregunta comparativa" que si se cumple, realizaría las instrucciones posteriores, mover variables, iniciar actuadores entre otras muchas acciones.

En esta función condicional hace una comparación con la variable *'tecla'* y con la asignación `NO_KEY`. Contiene un operador de 'no es igual que' (!=), este hace la siguiente comparación: Si tecla no es igual que NO_KEY (no se haya apretado tecla), por lo que este *'If'* dejará proseguir si la primera tecla es pulsada.

Cuando la condición sea afirmada, la variable *'tecla'* irá a la variable *'pos1'* y se mostrará en la pantalla la tecla pulsada, en su respectivo lugar.

```
if (tecla != NO_KEY)
// Condición que se utiliza para saber si se ha pulsado la tecla
{
  pos1 = (tecla);
// Guarda el valor tecla en la variable pos1
  lcd.setCursor(11,1);
  lcd.print(pos1);
```

Una vez se haya presionado una tecla, es necesario que el programa espere a la segunda tecla, para ello se utilizará otra vez el comando *waitForKey()* pero cambiando la variable *'tecla'* por *'tecla2'*.

```
  char tecla2=customKeypad.waitForKey();
  // Espera hasta pulsar la segunda tecla
  // Guarda valor pulsado en la variable tecla2
```

Estando dentro de la primera función condicional '*If*', el valor de la segunda tecla una vez esta sea pulsada se guardará en la variable '*tecla2*'. Sin dejar acabar la función condicional, se introduce la segunda función condicional, exactamente igual que la primera, pero esta vez para la variable '*tecla2*'.

```
    if (tecla2 != NO_KEY) // Condición que se utiliza para saber si se
ha pulsado la segunda   tecla
    {
      pos2 = (tecla2);   // Guarda el valor tecla2 en la variable pos2
      lcd.setCursor(11,3);
      lcd.print(tecla2);
      delay(1000);
```

Una vez que la segunda tecla sea pulsada se guardará en su variable correspondiente, y esta se mostrará en la pantalla LCD tal y como ocurre en el caso de la primera tecla.

El uso de un condicional dentro de otro condicional es como un tipo de método de seguridad, es decir, si la primera y segunda tecla son pulsadas, proseguirá con la línea de sucesos; mientras la primera no sea pulsada y/o la segunda tampoco sea pulsada, se mantendrá a la espera hasta ser pulsadas.

```
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.write("Presione '#'");
        lcd.setCursor(0,1);
        lcd.write("para autorizar");
        lcd.setCursor(0,2);
        lcd.write("movimientos:");
        lcd.setCursor(0,3);
        lcd.write("POS1:   , POS2:");
        lcd.setCursor(6,3);
        lcd.write(pos1);
        lcd.setCursor(16,3);
        lcd.write(pos2);
```

Cuando las dos teclas se hayan presionado, por seguridad se ha creído conveniente crear un sistema de autorización con otra tecla. Con los comandos de la pantalla LCD, limpiamos la pantalla y escribimos el texto para que el usuario lea que para autorizar los movimientos necesita apretar una tercera tecla.

La tercera tecla a presionar será el símbolo '#', se ha creído conveniente este símbolo ya que en el programa no tenía ningún uso, y así no habrían confusiones con los números. El sistema es muy sencillo, si se presiona una tecla y esta es el símbolo '#', seguirá actuando con las siguientes instrucciones, que estas ya serán los movimientos de los motores para cada una de las posiciones.

```
 char tecla3=customKeypad.waitForKey();
// Espera hasta pulsar la tercera tecla
// Guarda valor pulsado en la variable tecla3

      if (tecla3 == 35) //Condición para confirmar la aceptación de
ordenes
      {
      lcd.setCursor(13,2);
      lcd.print(tecla3);
```

Este es el tercer condicional, que está dentro del segundo y este dentro del primero, y así enlazarlos todos. Para esta condición se necesita el código decimal del símbolo '#' en ASCII, este es el 35 (DEC). La condición será, si el valor pulsado de la tercera tecla es 35 ('#'), entonces muestra la imagen y pasa a la fase de las funciones *Switch{}* y sus respectivos *Case{}*.

```
     else
      {

        lcd.clear();
        lcd.setCursor(5,1);
        lcd.write("MOVIMIENTO");
        lcd.setCursor(5,2);
        lcd.write("CANCELADO");
        delay(3000);

        lcd.clear();


      }
```

Por lo contrario, si esta condición no se cumple, borraría la pantalla LCD y mostraría 'Movimiento cancelado', una espera 3 segundos, borraría la pantalla de nuevo y reiniciaría el bucle para poder empezar el programa nuevamente. La instrucción *else* indica que la condición no ha sido cumplida y por lo tanto ha de realizar las acciones a continuación.

7.2.3.2. Programación de posiciones

El puente grúa tendrá un total de 9 posiciones, desde la posición inicial 0 hasta la posición 8. Para determinar las posiciones no se usarán *encoders* ni ningún otro sensor de posición. Se programarán las posiciones a partir de las instrucciones *switch*, es un método para seleccionar varias opciones discretas; cada opción irá precedida con su respectiva instrucción '*case*'.

El primer '*switch*' será el indicado para la primera tecla pulsada, es decir, para la primera posición. Esta será la primera instrucción principal: `switch(pos1)`, dentro del paréntesis encontramos la variable '*pos1*', esta es la encargada de guardar el valor obtenido al presionar la primera tecla. Cuando la instrucción '*switch*' reciba el valor de la primera tecla, pasará a la fase de selección del '*case*' indicado. Cada '*case*' tiene asignado un valor en decimal, por lo que sí coincide con el valor del '*switch*' ese será el seleccionado y por lo tanto se realizarán las instrucciones asignadas a ese '*case*'.

Hay dos tipos de '*switch*' el principal que asignará la primera posición y el secundario que será el de la segunda posición. A el principal le pertenecen un total de 12 '*case*', estos son: posiciones del 0-8, tecla numérica 9, teclas de símbolo # y *.

Mientras que para cada '*case*' de posición de la 1 a la 8 tienen un '*switch*' secundario que tiene otros 12 '*case*'. Esto se debe a que si se eligiese por ejemplo, primero la posición 2, y luego la posición 6, la primera procede a una posición y desde esa tiene que ir a la otra, nunca vuelve a un punto inicial para ir a la segunda posición. Eso quiere decir que hay un total de 72 movimientos diferentes (desde las posiciones 1 a 8 se pueden realizar 9 movimientos en cada una de ellas), y por lo tanto se necesitan tantos '*case*' como sean precisos.

A parte de las posiciones principales de la 1 a la 8, tenemos que tener en cuenta que pasaría si se pulsara por ejemplo un '#' o un '9' cómo primera tecla.

Si la tecla no fuese de un '1' a un '8', la segunda tecla no importaría cual fuera, ya que una vez las dos teclas pulsadas y la tercera para autorizar, el primer '*case*' se seleccionaría, se mostraría un texto en la pantalla LCD cómo por ejemplo: "Posición errónea". Este se mostraría durante un par de segundos y con la instrucción '*break*' saltaríamos el '*switch*' principal y al no haber ningún '*switch*' secundario, se acabaría este bucle para poder empezar otro de nuevo.

```
switch(pos1){

        case 35:  //Posición errónea (#)

        lcd.clear();
        lcd.setCursor(0,1);
        lcd.write("Posición errónea");
        lcd.setCursor(0,2);
        lcd.write("Inténtelo de nuevo");
        delay(2500);
        lcd.clear();
        break;

        case 42:  //Posición errónea (*)

        lcd.clear();
        lcd.setCursor(0,1);
        lcd.write("Posición errónea");
        lcd.setCursor(0,2);
        lcd.write("Inténtelo de nuevo");
        delay(2500);
        lcd.clear();
        break;


        case 48: //Posición inicial (0)

        lcd.display();
        lcd.clear();
        lcd.setCursor(0,1);
        lcd.write("Ya está en la");
        lcd.setCursor(0,2);
        lcd.write("posición inicial");
        delay(2500);
        lcd.clear();
        break;
```

```
case 57:  //Posición errónea (9)

lcd.clear();
lcd.setCursor(0,1);
lcd.write("Posición errónea");
lcd.setCursor(0,2);
lcd.write("Inténtelo de nuevo");
delay(2500);
lcd.clear();
break;
```

Una vez solventados los problemas con las teclas inservibles, hay que centrarse en la programación de las posiciones. Se explicará el '*case*' principal de la primera posición con todas sus respectivas posiciones secundarias. Es necesario tener en cuenta que los pasos establecidos para cada motor se tienen que haber calculado previamente. La distancia recorrida por el motor del eje X no es el mismo que el del eje Y, y por lo tanto cada motor necesita de su respectiva variable '*pasos*'.

Las secuencias siempre irán en el mismo orden, se moverá el eje Y su recorrido, en segundo lugar se moverá el eje X; posteriormente irá la secuencia del eje Z con la pinza, primero bajará el eje Z, cerrará la pinza y volverá a subir el eje Z. Esta sería una secuencia principal, en donde se recoge el objeto, y en la secundaria es donde se deja. La diferencia entre una y otra es que en la primera se cierra la pinza y en la segunda se abre.

A continuación se puede observar una secuencia principal, es el caso de la primera posición (tecla '1') y dentro de dicha secuencia otra instrucción de '*switch*' para sus movimientos secundarios.

```
case 49: //Primera posición

       myStepper2.step(-pasos2);    //motor eje y
       myStepper3.step(-pasos5);    //motor eje z
       myStepper4.step(-pasos6);     //cierra pinza
       myStepper3.step(pasos5);
```

```
      switch(pos2){
```

```
case 35:  //Posición errónea (#)
lcd.clear();
lcd.setCursor(0,1);
lcd.write("Posición errónea");
lcd.setCursor(0,2);
lcd.write("Inténtelo de nuevo");
myStepper3.step(-pasos5);
myStepper4.step(pasos6);     //abre pinza
myStepper3.step(pasos5);
myStepper2.step(pasos2);
delay(2500);
lcd.clear();
break;

case 42:  //Posición errónea (*)
lcd.clear();
lcd.setCursor(0,1);
lcd.write("Posición errónea");
lcd.setCursor(0,2);
lcd.write("Inténtelo de nuevo");
myStepper3.step(-pasos5);
myStepper4.step(pasos6);
myStepper3.step(pasos5);
myStepper2.step(pasos2);
delay(2500);
lcd.clear();
break;

case 57:  //Posición errónea
lcd.clear();
lcd.setCursor(0,1);
lcd.write("Posición errónea");
lcd.setCursor(0,2);
lcd.write("Inténtelo de nuevo");
myStepper3.step(-pasos5);
myStepper4.step(pasos6);
myStepper3.step(pasos5);
myStepper2.step(pasos2);
delay(2500);
lcd.clear();
break;
```

Si la primera tecla es un ‘1’ la placa controladora identificará que es la primera posición, pero, y si la segunda tecla es un: ‘#’, ‘*’ o bien ‘9’, para estos no hay posiciones determinadas. Para resolver este problema se basa en retroceder los pasos hechos hasta el momento, es decir, si la máquina va hacia la primera posición y recoge el objeto, se retrocede desde el último punto. Primero dejará en la misma posición el objeto, y luego retrocederá hasta la posición inicial.

Cuando la segunda tecla sea un '0', la secuencia será muy similar a las anteriores, ya que las otras vuelven al estado inicial igual que esta, solo cambiará el texto mostrado en la pantalla por "Volviendo a inicio" en cambio de "Posición errónea".

```
case 48:
lcd.clear();
lcd.setCursor(0,1);
lcd.write("Volviendo a inicio");
myStepper3.step(-pasos5);
myStepper4.step(pasos6);
myStepper3.step(pasos5);
myStepper2.step(pasos2);
break;
```

Las siguientes secuencias pertenecen a las 8 posiciones restantes desde la '1' hasta la '8'; en el caso de la primera la secuencia también será de retroceso, al estar dentro del '*case*' principal de la posición '1', la segunda secuencia en la posición '1' no tiene sentido, por lo tanto retrocederá sobre sus pasos hasta llegar a la posición inicial.

El resto son los movimientos necesarios para llegar a las posiciones indicadas en cada '*case*', estas secuencias están predeterminados por la primera posición.

```
case 49:  //P1
myStepper3.step(-pasos5);
myStepper4.step(pasos6);
myStepper3.step(pasos5);
myStepper2.step(pasos2);
break;

case 50:  //P2
myStepper.step(-pasos3);
myStepper3.step(-pasos5);
myStepper4.step(pasos6);
myStepper3.step(pasos5);
myStepper.step(pasos3);
myStepper2.step(pasos2);

break;

case 51:  //P3
myStepper.step(-pasos4);
myStepper3.step(-pasos5);
myStepper4.step(pasos6);
myStepper3.step(pasos5);
myStepper.step(pasos4);
myStepper2.step(pasos2);
break;
```

```
        case 52:  //P4
        myStepper2.step(pasos1);
        myStepper3.step(-pasos5);
        myStepper4.step(pasos6);
        myStepper3.step(pasos5);
        myStepper2.step(pasos1);
        break;

        case 53:  //P5
        myStepper.step(-pasos3);
        myStepper2.step(pasos1);
        myStepper3.step(-pasos5);
        myStepper4.step(pasos6);
        myStepper3.step(pasos5);
        myStepper.step(pasos3);
        myStepper2.step(pasos1);
        break;

        case 54:  //P6
        myStepper.step(-pasos4);
        myStepper2.step(pasos1);
        myStepper3.step(-pasos5);
        myStepper4.step(pasos6);
        myStepper3.step(pasos5);
        myStepper.step(pasos4);
        myStepper2.step(pasos1);
        break;

        case 55:  //P7
        myStepper.step(-pasos3);
        myStepper2.step(pasos2);
        myStepper3.step(-pasos5);
        myStepper4.step(pasos6);
        myStepper3.step(pasos5);
        myStepper.step(pasos3);
        break;

        case 56:  //P8
        myStepper.step(-pasos4);
        myStepper2.step(pasos2);
        myStepper3.step(-pasos5);
        myStepper4.step(pasos6);
        myStepper3.step(pasos5);
        myStepper.step(pasos4);
        break;


      }

    lcd.clear();
    break;
```

Este '*switch*' principal solo es de la primera posición, en los anexos hay el programa completo donde se encuentran todas las secuencias posibles. Para poder explicar bien y detalladamente esta parte del programa se ha creído conveniente poner solo un tramo de él. [8]

8. Diseño y construcción del prototipo

8.1. Introducción

El programa de automatización no podría comprobarse a no ser que haya un prototipo donde probarlo. Para ello se diseñará un prototipo, este tendrá las prestaciones necesarias para probar el uso del programa. El diseño del prototipo tendrá 5 partes importantes: eje X, eje Y, eje Z, pinza y por último el tablero de control. En este apartado se escogerán los materiales y piezas necesarias para realizar el proyecto así cómo realizar un presupuesto.

El diseño del prototipo en cuanto a dimensiones será acorde con el espacio disponible, por falta de espacio y otros inconvenientes económicos el autor del proyecto no ha podido dimensionarlo de otra manera. Así como los instrumentos, elementos de mecanización y otras piezas de construcción han sido adaptados para poder construir el prototipo. Algunas piezas serán diseñadas para su posterior impresión 3D, estas se desarrollaran con el programa Solid Edge. El diseño de un prototipo se puede idealizar, pero a efectos prácticos se puede encontrar uno con problemas a la hora de encontrar piezas, ya sea por un diseño raro de encontrar y muy específico o bien porque el coste de dicha pieza sería muy alto. Hoy en día gracias a las nuevas tecnologías, disponemos de impresoras 3D para la fabricación de piezas o conjunto de piezas para formar un artilugio. Gracias a esta máquina, es posible la creación de las piezas necesarias para el prototipo.

8.2. Elección de materiales

Un punto importante a destacar en el diseño de un prototipo es los materiales que se utilizará para su construcción. Es necesario dejar claro que este proyecto se basa en la automatización del prototipo, por lo que el desarrollo de la estructura y sus componentes mecánicos no se detallarán con gran precisión.

El material a elegir más importante es el responsable de la estructura en general y los ejes. Se debe tener en cuenta los pesos y fuerzas que tendrá que soportar la estructura principal, al ser un prototipo a escala, serán menores que las que soportaría el modelo real. Tanto la estructura como los ejes serán de aluminio por su resistencia y a la vez flexibilidad para poder amoldarse en las uniones. El aluminio elegido para el prototipo en general es: perfiles de 90º de 2 mm de grosor y perfiles rectos de 3 mm de grosor.

Los perfiles de aluminio adecuados para este proyecto tendrían que ser los de la ilustración 22.

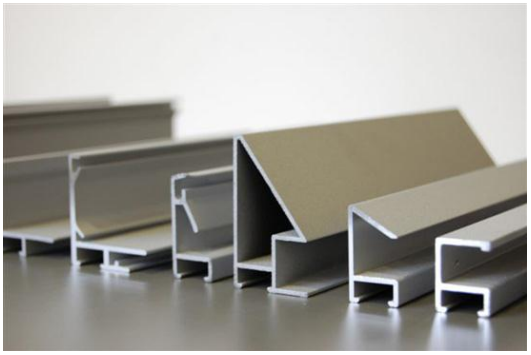


Ilustración 21 Perfiles de aluminio

Ilustración 22 Perfil modular de aluminio

Los perfiles modulares son los más habituales de encontrar en máquinas industriales, ya que su alta resistencia proporciona una gran estabilidad a la estructura. Además gracias a la forma del perfil se crean guías interiores, estas se podrían utilizar para encajar los ejes sin dificultad alguna. El único inconveniente de este tipo de perfil es su coste, este material se excede del presupuesto que el autor del proyecto ha acordado, por lo que se opta por construir el prototipo con materiales más económicos.

Tanto para la creación de las cubiertas de los mecanismos de los ejes X,Y y Z, como para el tablero de mandos, se utilizará un tipo de madera aglomerada llamada DM, por ser una madera de densidad media. Es perfecto para el uso que se le dará, una madera muy moldeable, que al cortarla dejará un acabado muy bueno. El DM elegido consiste en una plancha de 3mm de grosor con una densidad de 800 kg/m^3.

Ilustración 23 Planchas de DM (madera de densidad media)

El prototipo representará una habitación interior; como el DM no es viable utilizarlo en exteriores pero si en interiores, hace de esta madera un buen material para la construcción del prototipo. Para esta parte no se ha utilizado aluminio porque para cortar y doblar plancha de aluminio es bastante complicado si no se tienen las herramientas adecuadas.

Para la construcción de los raíles por donde se moverán los ejes se ha escogido también aluminio, pero esta vez de forma cilíndrica. Los tubos de aluminio podrán amoldarse perfectamente a los ejes y así conseguir la fricción necesaria para que el motor funcione a su pleno rendimiento.

Los tubos de aluminio que se utilizarán de railes tienen un diámetro exterior de 8mm con un grosor de 1mm. Son ligeramente flexibles, muy prácticos para amoldarse a las necesidades establecidas.

Ilustración 24 Tubos de aluminio

Otro elemento a destacar es la guía de movimiento de los ejes, esta servirá para la sujeción de los ejes y no dejar que se desvíen de su trayectoria. Proporcionando un movimiento recto y preciso. Para crear estas guías se utilizarán varillas de acero

inoxidable, dando una rigidez adecuada para una guía. Y para hacer el guiado con los ejes se utilizarán rodamientos lineales del mismo diámetro que la guía.

Ilustración 25 Varillas de acero inoxidable

Ilustración 26 Rodamientos lineales

Las varillas de acero inoxidable se han escogido de 8 mm de diámetro, ya que los rodamientos lineales más comunes que se encuentran en el mercado son de 8 mm de diámetro interior y por lo tanto el tamaño de las varillas se ha tenido que ajustar a los rodamientos encontrados en el mercado.

Por último, otro de los materiales más importantes del prototipo es el encargado de las transmisiones de movimiento. Se usarán varillas de acero roscadas, estas servirán para crear los ejes fijos de las ruedas o bien la creación de ejes transmisores de movimiento. Al ser varillas roscadas se podrán ajustar a las medidas necesarias con sencillas tuercas o bien tuercas autoblocantes según la necesidad. El tamaño de las varillas roscadas es de 5 mm de diámetro.

Ilustración 27 Varillas roscadas de acero

8.3. Estructura

Un buen diseño de la estructura garantizará un mejor funcionamiento de la máquina. Es necesario mencionar que, para ahorrar costes de construcción, el autor del proyecto ha creído conveniente utilizar material más económico, siguiendo el mismo diseño.

La estructura principal se basa en 4 puntos de apoyo a una base, estos puntos serán los perfiles de 90º de aluminio, estos tendrán que soportar todo el peso del prototipo. Los perfiles de 90º irán unidos entre sí gracias a los perfiles rectos, estos harán de soporte con la finalidad de dejar una estructura rígida. Los perfiles de 90º estarán clavados a una base de madera aglomerada. De esta manera se obtendrá una estructura sólida y rígida donde empezar a construir los ejes.

El diseño que se intenta simular es el de una habitación cuadrada, donde se instalaría el puente grúa automatizado.

Los soportes principales tendrán que aguantar el peso de los 3 ejes, estos llevan consigo los mecanismos de movimiento junto con los motores. El peso de los dos motores Nema 17 es de 640 gramos mientras que el de los dos 28byj-48 es de tan solo 64 gramos; en conjunto hacen aproximadamente 700 gramos. La estructura de los ejes sin contar los motores son de aproximadamente 2,7 quilogramos; contando los motores se llegaría a un total de 3,5 quilogramos aproximadamente. Es un peso muy relativo ya que se trata del prototipo, pero para la construcción de uno a escala real, el peso será una principal variable a tener en cuenta. Un peso de 3,5 quilogramos, es poco para una estructura sólida de aluminio, por lo que no tendría que suponer un problema.

Una vez se hayan ensamblado los perfiles y haya quedado una estructura bien rígida, se incorporarán los raíles por donde pasarán las ruedas del eje Y. Estos raíles son los tubos de aluminio de 8 mm de diámetro exterior. Se necesitan dos tubos cortados de manera que pase de un perfil al siguiente que tenga delante, formando así una estructura simétrica.

Cuando los raíles estén montados en la estructura, se podrá medir la altura del primer eje (eje Y) y así concretar la distancia que tendrá que haber entre el raíl y la varilla que hará de guía. La guía del eje Y será una varilla de acero inoxidable, sujeta del extremo de un perfil de 90º al otro que tendrá delante, formando una paralela con el tubo de aluminio. Esta irá sujeta con unas abrazaderas en cada extremo, pudiéndose montar y desmontar ante cualquier imprevisto.

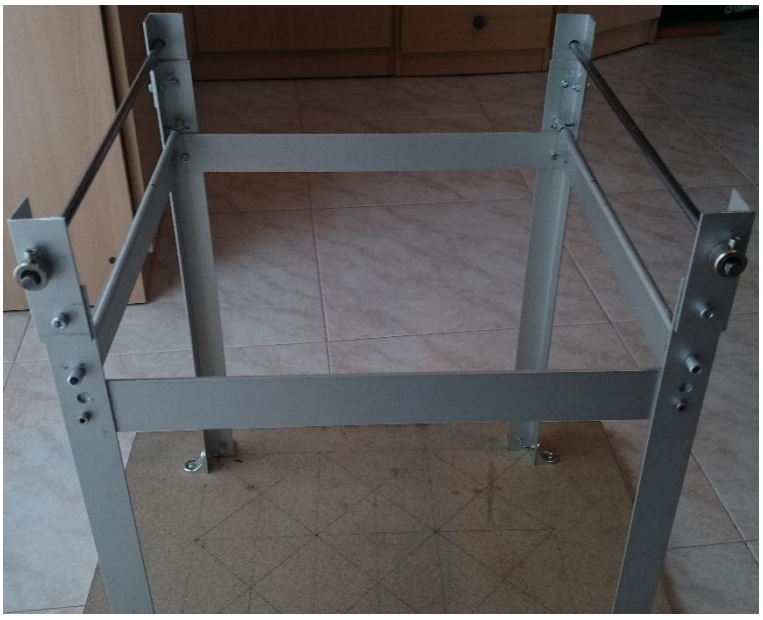


Ilustración 28 Estructura del prototipo

8.4. Ejes

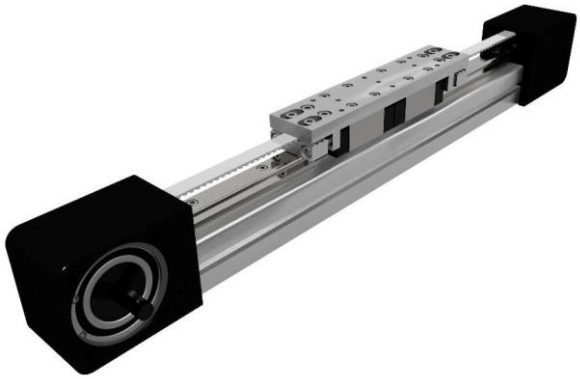
8.4.1. Transmisiones

Para un correcto movimiento del puente grúa, es necesario crear unos ejes donde en cada uno de ellos se instalará un mecanismo de movimiento. Hay múltiples formas de diseñar ejes de movimiento. Uno de los más efectivos en cuanto a precisión y soporte de fuerza es el de transmisión lineal por el giro de un tornillo de bolas deslizable, se muestra en la ilustración 29.

Este es uno de los métodos más efectivos para hacer una transmisión lineal. Para la construcción de los ejes en la realidad, tendría que ser este tipo de mecanismo, el inconveniente principal es su alto precio, ya que las piezas son de una precisión inigualable. Por lo que para el prototipo está descartado, el autor del proyecto cree necesario establecer costes más económicos.

Ilustración 29 Transmisión lineal por tornillo

Otra transmisión muy frecuente que se utiliza en desplazamiento lineal es por distribución de una correa dentada. Este mecanismo funciona muy parecido al de transmisión de tornillo, la transmisión por cadena dentada es muy útil para el uso en máquinas de precisión que el peso no es muy elevado.



Este mecanismo si se tuviese que comprar salía de presupuesto. Ya que los proveedores principales de estos mecanismos distribuyen a industrias y no directamente al consumidor. En la ilustración 30 se puede ver la transmisión por correa.

Ilustración 30 Transmisión lineal por cadena

Otro de los mecanismos que se tiene que tener en cuenta es la transmisión por rodamiento, como en los puente grúa. Esta transmisión es la más sencilla de todas y a la vez también muy efectiva dependiendo de las circunstancias, el eje Y y X de un puente grúa común tiene un mecanismo de este tipo.

En la ilustración 31 se puede observar el mecanismo de un puente grúa donde unas ruedas guiadas por un raíl se pueden desplazar por el movimiento generado por los motores. [13]

Ilustración 31 Puente grúa industrial

8.4.2. Eje Y

El primer eje a construir es el eje y, este está conectado directamente a los perfiles de aluminio que forman la estructura. El mecanismo escogido para este eje será el de transmisión por rodamiento; cogiendo la idea de un puente grúa, el eje Y se moverá gracias a unas ruedas colocadas sobre unos raíles, que conectadas a un motor paso a paso Nema 17 otorgará un movimiento lineal. Para mejor estabilidad del eje Y, se colocará una guía por encima de los raíles, que, con unos rodamientos lineales se creará un guiado con muy poco rozamiento.

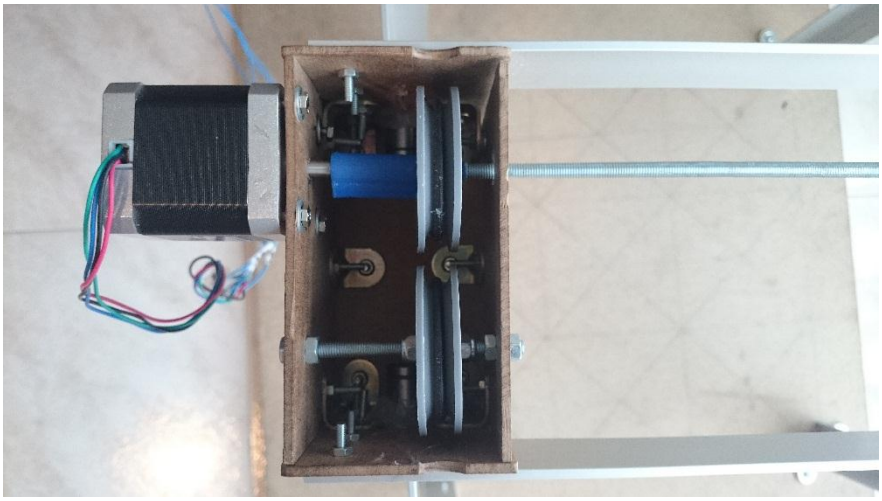


Ilustración 32 Mecanismo de transmisión del eje Y

En la ilustración 32 se puede observar el sistema de transmisión que tiene el eje Y. El motor Nema 17 va encajado mediante una pieza hecha a medida a una rueda de PVC con una junta tórica. También está roscada una varilla de acero a la pieza a medida, esta permitirá la transmisión del movimiento a la rueda del rail paralelo.

La varilla de acero roscada proporcionará al eje un movimiento lineal parejo, ya que cuando el motor ejerza la fuerza en una rueda también lo hará en su respectiva rueda paralela.

Ilustración 33 Estructura con el eje Y acoplado

En la ilustración 33 se observa el acople del eje Y a la estructura principal, el eje Y está guiado por los rodamientos lineales adjuntos en las cajas de transmisión, estas están unidas por dos perfiles rectos de aluminio. De esta forma el eje una vez acoplado a la estructura, forma un mecanismo robusto capaz de transportar mucho peso.

8.4.3. Eje X

El segundo eje a construir será este, el eje X, que irá situado encima del eje Y, pudiéndose transportar cuando este último se mueva. Cuando el eje Y se desplace por la estructura, el eje X podrá desplazarse por encima del eje Y. La construcción de este segundo eje es algo más sencilla, para ello se ha creído conveniente utilizar el mecanismo de transmisión por correa dentada. Al no tener que transportar tanto peso como es el caso del eje Y, un sistema por transmisión de correa es el más idóneo.

El motor Nema 17 se colocará encima de una de las cajas de transmisión del eje Y, este estará en posición vertical, para acoplar el engranaje que hará girar la correa. En la otra caja de transmisión estará el soporte donde se aguantará la correa.

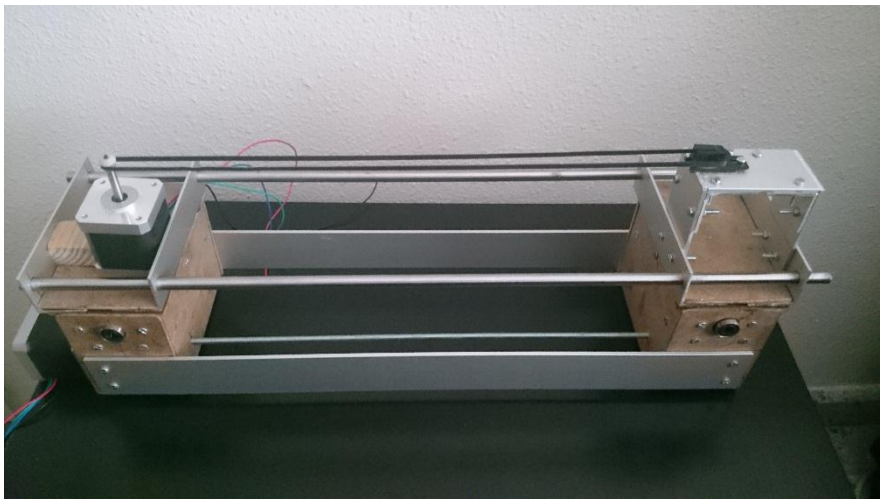


Ilustración 34 Mecanismo de correa del eje X acoplado al eje Y

Una vez que la correa esté unida se deberá crear otra caja de transmisión, donde se encajaran dos puntos de la correa. Cuando el motor gire y la correa se desplace, la caja de transmisión también se moverá. Para un mejor movimiento y no tener problemas con el peso, se añadirán al eje X unas guías como en el eje Y. Unas varillas de acero inoxidable que con la ayuda de los rodamientos lineales, la caja de transmisión se deslizara con muy poco rozamiento, con lo que se eludirá el peso de la caja y por lo tanto se conseguirá un movimiento preciso.

8.4.4. Eje Z

Una vez acoplado el eje X en el eje Y, hay que construir el eje Z para poder incluirlo dentro de la caja de transmisiones del eje X. Este permitirá moverse por el espacio vertical, pudiendo subir y bajar la pinza. El mecanismo de transmisión pensado para este eje es el de transmisión de piñón-cremallera, el eje Z se ha diseñado para que al contrario de un puente grúa este sea rígido. Para poder desplazar el eje hacia arriba o abajo, es necesario que el propio eje esté dentado, dentro de la caja de transmisiones del eje X habrá un motor y a este se acoplará un engranaje, que mediante rodamiento desplazará el eje hacia arriba o abajo. En la ilustración 36 se puede ver un ejemplo de esta transmisión.

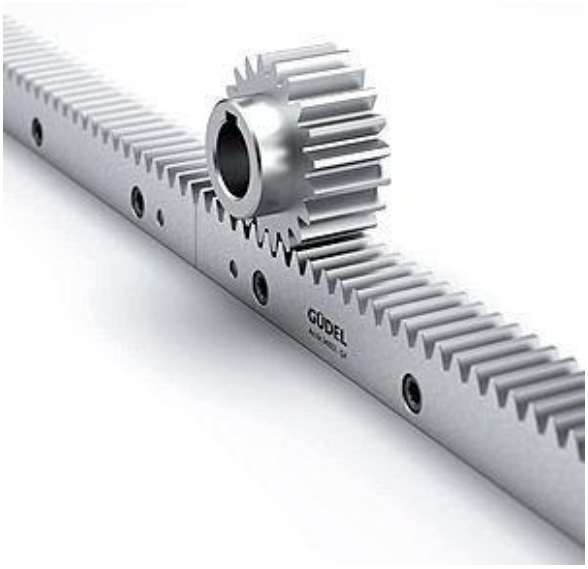


Ilustración 35 Transmisión piñón-cremallera

Al ser un eje fijo, otorgará una precisión ideal para el puente grúa que se quiere diseñar. En un puente grúa habitual, se encontrarían unos cables con un gancho acoplado, que mediante un sistema de poleas haría subir o bajar el gancho. Este sistema es muy útil cuando no se requiere precisión.

De esta manera, con un eje fijo, se puede saber la posición exacta que quedaría el extremo del eje, al final del eje irá acoplada la pinza.

8.5. Pinza

Uno de los objetivos principales de la creación del prototipo es que el puente grúa sea capaz de recoger un objeto en una posición y dejarla en otra. Para ello es necesario un sistema de agarre, ya sea un gancho, una pinza, u otro sistema. El autor del proyecto le ha parecido conveniente hacer el sistema de pinza, ya que sería el más práctico y a la vez el más preciso.

El diseño de la pinza se ha extraído de la web, el creador dejó el link de descarga [10]. Es un diseño que necesita unos cambios para que se pueda adaptar a las necesidades. Para ello se ha utilizado el programa Solid Edge ST7, los principales cambios son:

- Adecuar la base para el motor paso a paso.
- Cambiar el tipo de encaje para poder acoplar el motor paso a paso
- Cambios del diseño estético

La base de los engranajes tiene una cavidad para acoplar un micro-servomotor, para el caso de este proyecto, esta cavidad se tendría que suprimir al igual que los múltiples agujeros de soporte del servomotor. Utilizando la función '*extruir*' se pueden rellenar los agujeros y tapar la cavidad del servomotor. El resultado final es el de la ilustración 37, mientras que en la ilustración 36 es el original.

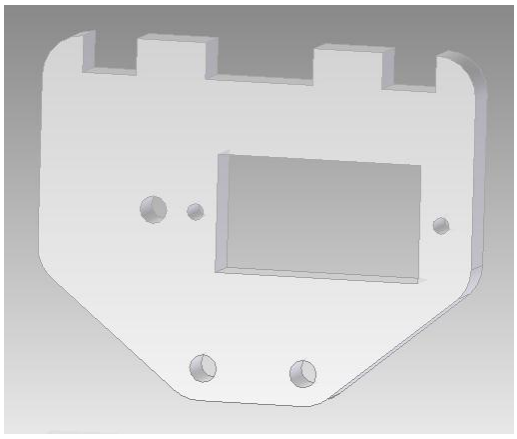


Ilustración 36 Pieza de soporte de engranajes (original)

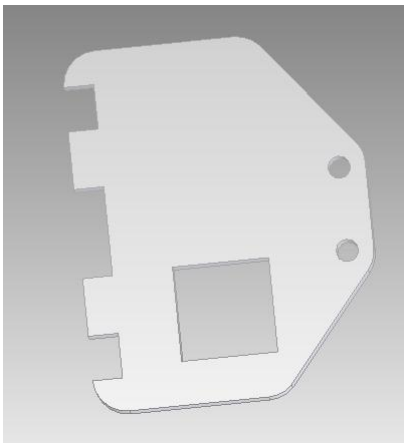


Ilustración 37 Pieza de soporte de engranajes (modificada)

Para dar movimiento a la pinza es necesario acoplar algún tipo de motor, en el caso del original iba acoplado un servomotor. En este proyecto en cambio, se usará un motor paso a paso para proporcionar el movimiento a la pinza, este motor paso a paso es el 28byj-48. El eje central de este motor es rectangular, mientras que el eje de transmisión del servomotor es circular con estrías. Para poder acoplar el motor paso a paso se ha tenido que cambiar la pieza del soporte, esta se ha dado una forma rectangular como anclaje, además se ha incrementado la altura de la pieza, para que encaje mejor el motor.

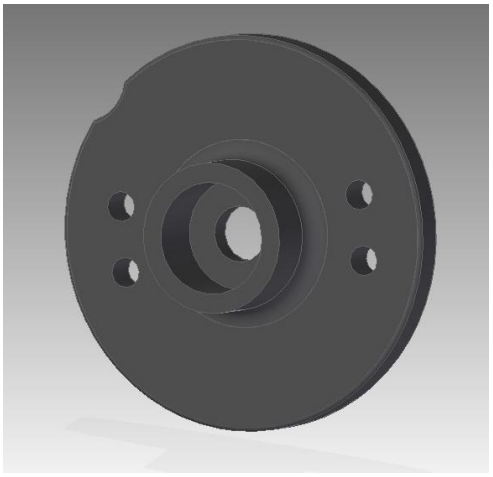


Ilustración 38 Pieza encaje de servomotor (original)

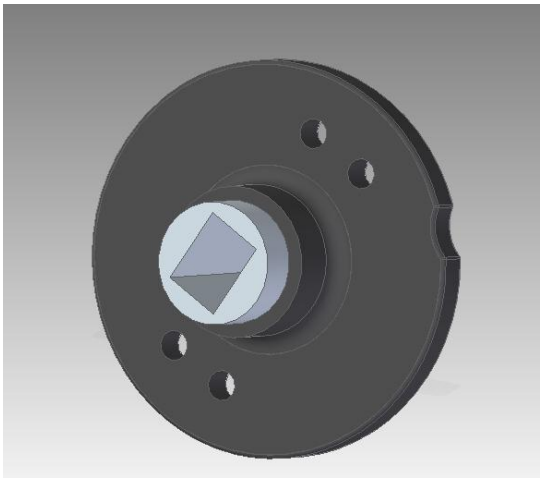


Ilustración 39 Pieza encaje de motor paso a paso 28byj-48 (modificada)

Los otros cambios que se han hecho en el conjunto de la pinza son modificaciones de diseño, no tienen una funcionalidad específica. Son cambios estéticos o bien para poder acoplar de manera correcta la pinza al resto del prototipo.

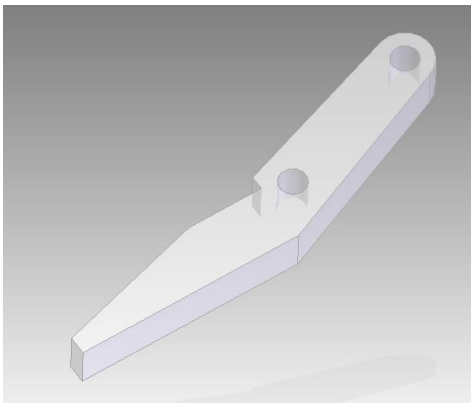


Ilustración 40 Pieza de una parte de la pinza (modificada)

Por ejemplo, la pieza de la pinza se ha rediseñado con una estética más atractiva, esta se muestra en la ilustración 40.

Finalmente, una vez recopiladas todas las partes, se juntarán para formar el diseño completo de la pinza. En la ilustración 41 se puede observar el diseño final de la pinza, y en la ilustración 42 se muestra el despiece de la pinza impresa en ABS. [14]

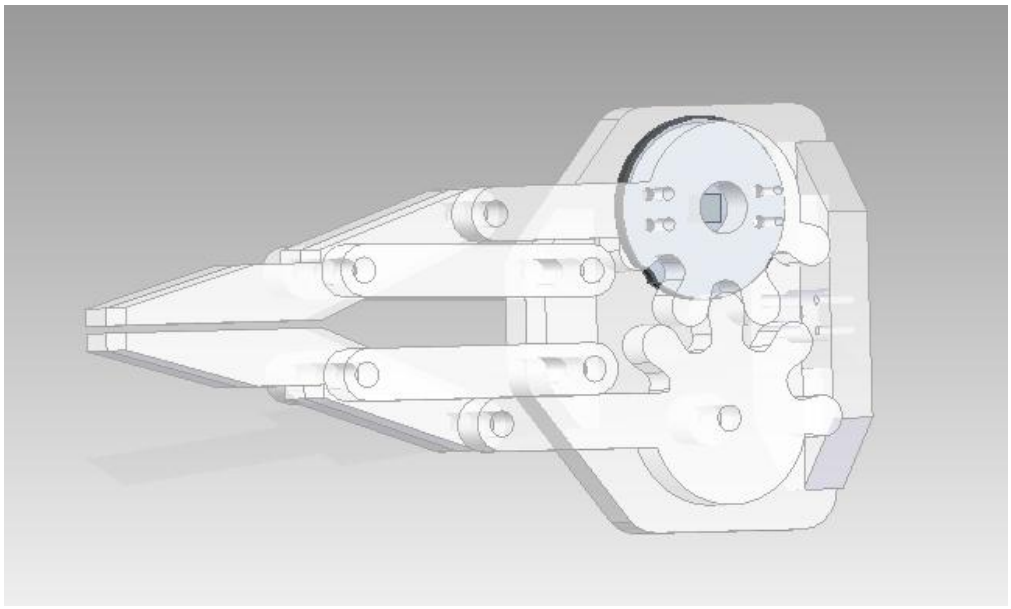


Ilustración 41 Diseño final de la pinza

Ilustración 42 Piezas de la pinza impresas en ABS

8.6. Impacto medioambiental

En este proyecto se tiene en cuenta el impacto medioambiental. Se intenta hacer uso de materiales no contaminantes en medida de lo posible.

El uso del prototipo es únicamente de interiores, no obstante también se tienen que tener en cuenta los materiales por impacto medioambiental. Los materiales principales utilizados en la elaboración del prototipo son:

- Aluminio, un material totalmente reciclable, incluso el reciclaje del aluminio es mucho más económico que la propia producción.
- Madera, el uso de aglomerados genera normalmente serrín y pequeñas astillas, todos estos residuos se pueden reciclar o bien aprovechar para otros usos más específicos.
- ABS, el material para crear piezas con la impresora 3D. Los residuos generados al fabricar las piezas, se guardan para su posterior reciclaje. Estos residuos se tienen que llevar a empresas externas que se encargarán de su reciclaje.
- Elementos electrónicos, cables u otros instrumentos, la mayoría de componentes electrónicos son perjudiciales para el medioambiente, una vez haya acabado su vida útil se tiene que llevar a lugares especializados para su correcta destrucción.

Una vez haya acabado la vida útil del prototipo se tendría que desmontar y ordenar el despiece por materiales, y estos llevarlos a lugares específicos de reciclaje o para su posterior destrucción.

8.7. Presupuesto

Artículo	Descripción	Ud	Precio/ud	Precio
Arduino Mega 2560 R3	Placa microprocesador	1	39,00€	39,00€
Cableado Arduino	Cables de conexiones Arduino	40	0,05€	2,00€
Display HD44780	Pantalla LCD para Arduino	1	18,00€	18,00€
Flexible KeyPad	Teclado matricial flexible	1	4,50€	4,50€
Nema 17	Motor paso a paso Nema 17	2	22,50€	45,00€
28BYJ-48	Motor paso a paso 28BYJ-48	2	6,50€	13,00€
L298	Driver de potencia para motor	2	3,50€	7,00€
ULN2003	Driver de potencia para motor	2	1,50€	3,00€
Cableado múltiple	Cableado para diferentes conexiones	8	0,30€/m	2,40€
Transformador 12v	Transformador para Arduino	1	20,00€	20,00€
Transformador 5v	Transformador para motores	2	12,00€	24,00€
Resistencias	Resist. Para el circuito de potencia	2	0,05€	0,10€
Potenciómetro	Pote. Para circuito de potencia	1	0,45€	0,45€
Base de madera	Madera aglomerada 60x60x2cm	1	8,00€/m^2	4,80€
Planchas de DM	Plancha de madera DM 50x50cm	2	4,50€/m^2	4,50€
Perfil aluminio	Perfil de aluminio de 1,80m rectos	2	7,00€	14,00€
Perfil aluminio	Perfil de aluminio de 1,80 de 90º	2	9,50€	19,00€
Varilla acero inox.	Varilla de acero inox. 2m 8mm Ø	1	7,25€/m	14,50€
Varilla roscada	Varilla de acero roscada 1m 5mm Ø	1	2,60€/m	2,60€
Rodamientos	Rodamientos lineales 8mm Ø	8	3,00€	24,00€
Poleas	Poleas de transmisión	4	4,00€	16,00€
Sistema correa	Correa con engranajes	1	12,00€	12,00€
Sistema cremallera	Cremallera con piñón	1	15,00€	15,00€
Tornillos y turecas	Tornillos y tuercas varias	--	15,00€	15,00€
Filamento ABS	Plástico para impresora 3D	1	22,00€/kg	22,00€
Total (sin IVA)				**341,85€**
Total (IVA 21% incluido)				**413,63€**

9. Posibles mejoras

Este proyecto se ha centrado en el diseño electrónico de un puente grúa de precisión, el diseño mecánico se basa en el prototipo. Una de las posibles mejoras para este proyecto sería crear un sistema mecánico mucho más eficiente y mejor diseñado, sería el claro ejemplo de un nuevo diseño en los ejes o en la pinza, así como en la estructura general.

Otra posible mejora o ampliación del proyecto sería conectar unos sensores de presión a la pinza, de esta manera se conseguiría controlar la presión que ejerce la pinza, y poder modularla dependiendo de las necesidades. Esta ampliación de proyecto, constaría de su parte electrónica ya que interviene programación de *software* y otros elementos de *hardware*.

Una de las partes que no se ha podido lograr en este proyecto es la conexión *Bluetooth*, era uno de los objetivos secundarios, pero para ajustarse a los límites de entrega no se ha podido incluir en el proyecto. Esta sería otra mejora, crear una aplicación *Android* para poder comunicarse vía *Bluetooth* y así mediante un *smartphone* poder controlar el prototipo.

10. Conclusiones

Finalmente, se puede decir que el proyecto se ha acabado con un cierto grado de éxito, ya que no se ha acabado tal y como se esperaba. La parte electrónica se ha completado junto con la programación del sistema de control, pero por falta de experiencia en el tema de maquetación combinada con la falta de tiempo, el prototipo no se ha construido tal y como se esperaba desde el principio. También es necesario comentar el elevado coste de algunos componentes, ya que eso ha impedido dejar un mejor acabado de la maqueta.

Estos problemas han ido surgiendo mientras se ha ido construyendo el prototipo, ya que me basaba en ideas teóricas. Con estos imprevistos uno se da cuenta que la teoría no es del todo precisa y siempre puede salir algún problema inesperado.

Los conocimientos aprendidos en este proyecto son muy gratos y reconfortantes, el hecho de buscar información para diseñar y construir la maqueta desde cero, y una vez obtenida, aplicarla, y ver que el proyecto va avanzando poco a poco, es muy gratificante. Además todos los conocimientos aprendidos, servirán para en un futuro conseguir hacer mejores proyectos y sobretodo poder aplicarlos en la vida laboral.

Por último, aun no consiguiendo realizar el proyecto al 100% de lo que se pretendía desde un principio, decir que estoy muy satisfecho de haber realizado este proyecto como meta personal.

11. Bibliografía

[1] Arduino official web page *consultado* 10/11/2014. URL http://arduino.cc/en/Main/arduinoBoardMega2560

[2] Luis Llamas web page *consultado* 12/11/2014. URL http://www.luisllamas.es/2014/04/arduino-puerto-serie/

[3] Arduino official web page *consultado* 12/11/2014. URL http://www.arduino.cc/en/Tutorial/LiquidCrystal

[4] Gzalo web page *consultado* 12/11/2014. URL http://gzalo.com/lcdalfanumerico/

[5] Arduino official web page *consultado* 13/11/2014. URL http://playground.arduino.cc/Main/KeypadTutorial

[6] 42Bots web page *consultado* 15/12/2014. URL http://42bots.com/tutorials/28byj-48-stepper-motor-with-uln2003-driver-and-arduino-uno/

[7] ElectronicLab tutoriales *consultado* 18/02/2015. URL http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/

[8] Arduino official web page *consultado* 19/02/2015. URL http://www.arduino.cc/en/reference/stepper

[9] Google Wikipedia *consultado* 23/10/2014. URL http://es.wikipedia.org/wiki/Processing

[10] Arduino official web page *consultado* 27/10/2014. URL http://playground.arduino.cc/ArduinoNotebookTraduccion/Structure

[11] Domótica Arduino web page *consultado* 4/11/2014. URL http://domotica-arduino.es/circuito-antirrebote-y-filtro-de-ruido-para-arduino-o-microcontrolador/

[12] Google Wikipedia *consultado* 17/03/2015. URL http://es.wikipedia.org/wiki/Puente-gr%C3%BAa

[13] Ingemecánica tutoriales *consultado* 9/03/2015. URL http://ingemecanica.com/tutorialsemanal/tutorialn121.html

[14] Archivo micro pinza Solid Edge de la universidad UEPG *consultado* 9/04/2015. URL http://deinfo.uepg.br/~ari/aulas/robotica/MicroGarra.rar http://www.uepg.br/